Hochschule Konstanz
Technik, Wirtschaft und Gestaltung

Praxissemester bei isys vision GmbH & Co. KG

Bericht und Erfahrungen

Lorenz Bung `lorenz.bung@htwg-konstanz.de`
HTWG Konstanz
Angewandte Informatik, 4. Semester

01. März 2018 bis 31. Juli 2018

Zusammenfassung

In diesem Bericht geht es um mein Praktikum bei isys vision GmbH & Co. KG, welches ich im Rahmen meines Bachelorstudiums im Fach Angewandte Informatik im 4. Studiensemester absolviert habe.

Ich werde beschreiben und erklären, welche Aufgaben mir weshalb gegeben wurden, wie ich sie gelöst habe und welche Werkzeuge und Vorgehensweisen ich dabei verwendet habe. Weiterhin werde ich die firmeninternen Abläufe, Ziele und Methoden nennen. Die mir während dem Praktikum zugeteilten Aufgaben umfassten ein breites Spektrum mit dem Fokus Webentwicklung und Javascript. Dabei war ich in mehreren Bereichen der Firma tätig und bekam so ein umfassendes Bild der Arbeit.

Meine Tätigkeit handelte von der Modifikation eines bereits bestehenden Webinterfaces, der Einrichtung eines Datenbankservers und eines Projektmanagementsystems sowie dem Aufbau eines Steuerungstools zur Kontrolle eines Roboters und einer WebGL-basierte Browseranwendung zur Vorschau von Punktwolken.

Abschließend werde ich ein Fazit aus den gesammelten Erfahrungen ziehen und positive bzw. negative Aspekte meiner Tätigkeit hervorheben. Ich werde außerdem zusammenfassen, inwiefern das Praxissemester mich weiter gebracht hat und mein akademisches Profil durch praktische Erfahrung abrundet.

Inhaltsverzeichnis

0 Vorwort **1**

1 Web-Oberfläche für das IVS **2**

1.1 Einlesen in vorhandenen Code und Redesign 3

1.1.1 Flat-Design mithilfe von CSS 3

1.1.2 Javascript-Framework 4

1.2 Dokumentation des Frameworks 4

1.2.1 Codekommentierung mit JSDoc 5

1.2.2 Beschreibung der API 5

2 Integration von Jira und Confluence **7**

2.1 Einrichtung der Datenbank 8

2.2 Einrichtung des Intranetservers 9

2.3 Ausfallsicherheit über ein RAID-System 10

3 Web-Tools für das Mikado-Projekt **12**

3.1 Demoablauf für Messen 12

3.2 Web-Remote für die Mikado-Software 14

3.2.1 Design der neuen Webseite 14

3.2.2 Befehls- und Logübertragung mithilfe von RosJS 18

3.2.3 Webhosting und Netzwerk 20

3.3 Web-Preview zur Darstellung von Punktwolken 21

3.3.1 Rendering mit `THREE.js` 22

3.3.2 Konfiguration durch JSON-Dateien 24

4 Persönliche Bilanz und Ausblick **25**

Listings

1.1 Ins HTML eingebettete Befehle für das IVS 5
2.1 SQL-Befehle zur Erstellung der Datenbanken 8
2.2 Änderungen in `/etc/network/interfaces` 10
3.1 Struktur der Kontrollseite 15
3.2 Bildelemente in einer SVG-Grafik 16
3.3 Serviceaufruf durch RoslibJS 19
3.4 Python-Skript zum Hosten des Webservers 20
3.5 Initialisierung von `THREE.js` 22
3.6 Inhalt von `index.json` 24

Abbildungsverzeichnis

1.1 Interface vor dem Redesign 4
1.2 Interface nach dem Redesign 4

3.1 Mikado-Roboter von Misubishi 12
3.2 Logfenster 18
3.3 Kontrolle 18
3.4 Info / Menü 18
3.5 Punktwolke mit Beispielfunden in der Webvorschau 22

Tabellenverzeichnis

3.1 Vor- bzw. Nachteile einer höheren vs. niedrigeren Mindestqualität 13

0. Vorwort

Zu Beginn möchte ich kurz die Hintergründe meines Praktikums erläutern, z.B. einige Informationen zu isys vision, die Wahl der Firma und den Bewerbungsprozess.

isys vision[1] ist ein Softwarehersteller, der sich mit grafischen Lösungen im industriellen Einsatz auseinandersetzt. Die Bildverarbeitung steht hierbei im Vordergrund und wird durch die von isys produzierte Software durchgeführt. Es handelt sich hierbei um ein mittelständisches Unternehmen mit etwa 20 Mitarbeitern. Die im selben Gebäude gelegene Tochterfirma Ensenso[2] stellt Kamera- und Hardwarekomponenten her, welche dann in Kombination mit der Software von isys vision als Gesamtpaket an Endkunden, aber auch Zulieferer und Großkunden verkauft wird. Dies umfasst hauptsächlich Komponenten wie Kameras, Beleuchtungsbauteile und Verbindungskabel sowie vollständige Rechensysteme, als auch die entwickelte Software.

Schon im Grundstudium war mir klar, dass ich für mein praktisches Studiensemester Firmen im Bereich der Bildverarbeitung, -generierung oder aber im Webumfeld suchen werde. Mit einem Praktikum in genannten Gebieten wollte ich einerseits meine Entscheidung für die Vertiefungsrichtung der Medieninformatik im Hauptstudium bekräftigen, andererseits war dies schon immer ein Bereich der Informatik, welcher mich besonders gereizt hat. Aus diesen Gründen habe ich mich unter Anderem bei isys vision beworben und schlussendlich ein halbes Jahr hier verbracht.

[1] http://www.isys-vision.de
[2] http://www.ensenso.com

1. Web-Oberfläche für das IVS

Das erste größere Projekt war die Entwicklung einer Web-Oberfläche für ein bereits bestehendes, jedoch noch nicht vollständiges Bildverarbeitungssystem. Sowohl Webinterface als auch Bildverarbeitung waren bereits vorhanden, jedoch nicht für den aktuellen Anwendungszweck geeignet und somit war eine Überarbeitung unabdingbar.

Die Aufgabe des betreffenden Systems ist die Ausrichtung einzelner Keramikseiten, welche dann in weiteren Schritten verarbeitet werden. Dabei geht es um die Produktion von Lambdasonden, wie sie in der Abgasfilterung von Autos zum Einsatz kommen.

Diese Keramik-"sheets" werden dabei mit hochpräziser Genauigkeit ausgestanzt, weswegen sie vor dem Stanzen entsprechend ausgerichtet werden müssen. Eine physikalische Ausrichtung ist dabei so gut wie unmöglich, da die gewünschte Genauigkeit dabei nicht erreicht werden kann.

Das von isys entwickelte System erkennt nun mithilfe zweier Kameras die Lage zweier Marken auf dem Sheet und somit die Lage auf dem verschieb- und rotierbaren Tisch. Mithilfe der Bildverarbeitungssoftware wird nun eine Motorbewegung errechnet, welche eine Ausrichtung des Werkstücks bewirkt. Anschließend kann das Teil durch weitere Maschinen in seiner nun festgelegten Position weiterverarbeitet werden.

Die Software besteht dabei nun aus drei verschiedenen Komponenten:

- Dem *ScbDrv*, welcher für die Ansteuerung der Motoren und Hardware verantwortlich ist.
- Dem *IVS* (Isys Vision Server), welcher für die Auswertung der Kameradaten sowie die Bildverarbeitung und Errechnung der Endposition verantwortlich ist.
- Dem *Webinterface*, welches dem Endkunden eine Möglichkeit zur Einsicht des aktuellen Status sowie zur Kontrolle über die Maschine gibt.

Während der ScbDrv nur geringfügig an die neue Maschinenkonfiguration angepasst werden musste, war mein Betreuer mit der Anpassung des IVS an die neuen Marken (in diesem Fall die Ecken der Keramiksheets) beschäftigt. Meine Aufgabe bestand aus dem visuellen Redesign der

Weboberfläche, dem Anpassen des Inhaltes (Entfernen überflüssiger Elemente, Hinzufügen von neuen, wichtigen Dingen) sowie der Anpassung der Funktionalität mithilfe von HTML, CSS und JavaScript.

1.1 Einlesen in vorhandenen Code und Redesign

Da das Interface auf der alten, bereits vorhandenen Version aufbauen sollte, war das Einarbeiten in bereits vorhandenen Code notwendig. Um Änderungen vorzunehmen war dabei ein großes Verständnis für diesen Code nötig, da es sich um komplexe und vor allem abstrakte Abläufe handelte, die nicht leicht nachzuvollziehen waren.

1.1.1 Flat-Design mithilfe von CSS

Da meine bisherigen Erfahrungen mit Javascript und jQuery mittelmäßig bis wenig vorhanden waren, machte ich mir zunächst das grafische Redesign der Webseite zur Aufgabe. Mit CSS hatte ich bereits gearbeitet, sodass ich mich verhältnismäßig schnell in das vorhandene Stylesheet einarbeiten konnte und erste Änderungen vornahm.

Dazu gehörten beispielsweise das Entfernen von Schatten (`box-shadow`), die Anpassung von Farben oder das ersetzen von Hintergrundbildern. Durch bereits wenige Änderungen im Stylesheet konnte ich hierbei recht große Erfolge erzielen, was die Modernität der Seite angeht.

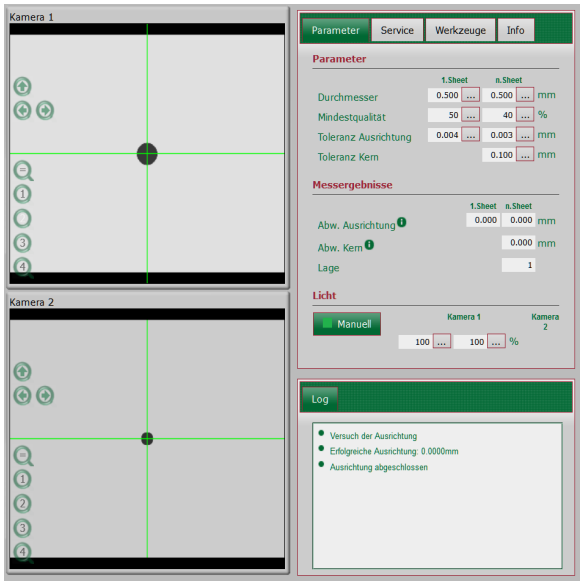

Abbildung 1.1: Interface vor dem Redesign

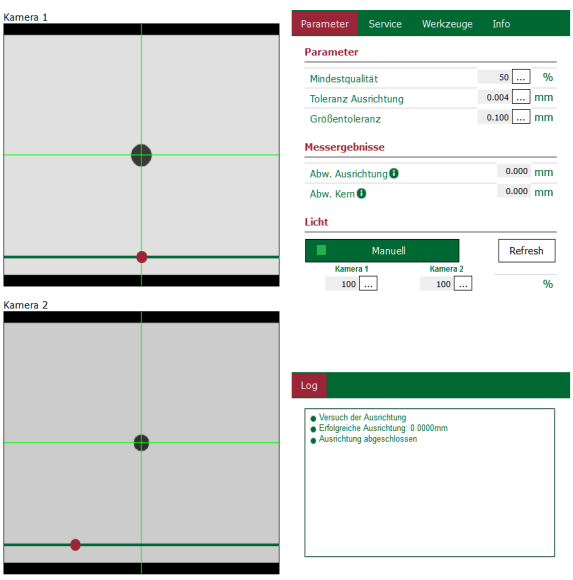

Abbildung 1.2: Interface nach dem Redesign

1.1.2 Javascript-Framework

In der bereits vorhandenen Version der Webseite wurde sämtliche Funktionalität durch ein Javascript-Framework geregelt, welches unabhängig von Inhalt und Design steht. Im Folgenden ging es nun darum, den hier geschriebenen Code durchzugehen, zu verinnerlichen und zu verstehen.

Aufgrund der trotz weniger Codezeilen doch sehr hohen Komplexität und vor allem den fehlenden Kommentaren gestaltete es sich als sehr schwierig, sich einzuarbeiten. Ein weiterer Nachteil war meine mangelnde Erfahrung mit Javascript; durch viel Recherche und weiterführende Beispiele konnte ich mir jedoch einiges herleiten und somit verstehen.

Weitere Modifikationen am Framework umfassten vor allem die Vereinfachung und Lesbarkeit des Codes. Die Hauptaufgabe in dieser Phase war das Verständnis, da dies die Grundlage für weitere Arbeit am Javascript war.

1.2 Dokumentation des Frameworks

Eine weitere wichtige Aufgabe war das Dokumentieren des bereits vorhandenen Codes. Da der ursprüngliche Autor leider nicht mehr bei isys vision angestellt war und ich mich nun schon länger

damit auseinander gesetzt hatte, sollte nun eine umfassende Dokumentation des Aufbaus erstellt werden.

Dies erleichtert einerseits die Arbeit am Framework selbst (Änderungen und Updates beispielsweise) und bietet denjenigen, die es nur verwenden (z.B. für eine neue Webseite mit demselben Framework) ein einfaches, gut verständliches Interface. Weiterhin können komplexere Vorgänge wie die Datenübertragung anschaulich erklärt und mithilfe von Diagrammen verdeutlicht werden.

1.2.1 Codekommentierung mit JSDoc

Im ersten Schritt arbeitete ich dabei nur am Quelltext selbst, den ich zuvor schon in lesbare Form gebracht hatte. Durch zusätzliche Kommentare an wichtigen und irreführenden Stellen konnte ich dabei eine erste Grundlage schaffen. Die größte Änderung war die Einführung einer Codedokumentation im `JSDoc`-Format[3], ähnlich zu den bei Java eingesetzten JavaDoc-Kommentaren. Die Vorteile davon liegen auf der Hand: Jede Methode bzw. Funktion hat somit eine eigene Beschreibung, in der Funktionsweise, Parameter, Rückgabewert usw. beschrieben werden.

Ein weiterer Vorteil von JSDoc ist die Möglichkeit, die Dokumentation automatisch zu generieren. Dies ist mit dem JSDoc-Tool von `Node.js` möglich, was nach der Installation einfach über den Befehl `jsdoc file.js` aufgerufen kann. Somit wird ein gut strukturiertes Dokument (inklusive Referenzen) automatisch erstellt.

1.2.2 Beschreibung der API

Nachdem der Quellcode selbst dokumentiert war, konnte ich mich nun mit den internen Abläufen des Frameworks beschäftigen und diese anhand von Grafiken anschaulich machen. In diesem Fall ging es vor allem um die Daten- und Befehlsübertragung vom IVS zum Webinterface und umgekehrt.

Die Übertragung der Daten erfolgt hierbei über eine `XMLHttpRequest`, bzw. (bei älteren Browsern) über ein `ActiveXObject`. Die Befehlscodes sind dabei im HTML-Code der betreffenden Buttons gespeichert:

```
1 <input type="text" size="6" maxlength="6" name="lms/SheetF.MinQuality">
```

[3] `https://en.wikipedia.org/wiki/JSDoc`

```
2 <button name="lms/Cmd" value="10">Ausrichtung</button>
3 <button name="lms/Cmd" value="20">
4   <ul>
5     <li>LmsScb/Mot.RefOk</li>
6     <li>Referenzfahrt</li>
7   </ul>
8 </button>
```

Listing 1.1: Ins HTML eingebettete Befehle für das IVS

In Zeile 1 wird beispielsweise das Register `lms/SheetF.MinQuality` geschrieben und auf einen sechsstelligen Wert gesetzt, der vom Nutzer eingegeben wird.
In Zeile 2 befindet sich ein einfacher Befehlsbutton, der den Wert `10` ins Register `lms/Cmd` schreibt (beim IVS ist dies der Code für eine Ausrichtung).
In den Zeilen 3 - 8 findet sich nun ein Knopf mit Status, der

- den Wert `20` ins Register `lms/Cmd` schreibt und somit eine Referenzfahrt auslöst
- innerhalb der unsortierten Liste (Zeile 5) den Wert `LmsScb/Mot.RefOk` liest (Status der Referenzfahrt).

Die Liste (`<ul>`) dient dabei ausschließlich der Darstellung und hat keine Auswirkung auf die Funktion; die Listenelemente werden innerhalb des Buttons entsprechend formatiert. Weiterhin wird je nachdem, was im gelesenen Register `LmsScb/Mot.RefOk` steht, ein entsprechendes Label gesetzt (grün falls Referenzfahrt OK, rot falls nicht).

Im Javascript-Framework wird nun ein Event-Listener für alle Buttons und Eingabefelder erstellt, der dann die entsprechenden Befehle erkennt. Diese werden in Form einer URL in einer Liste gespeichert, welche dann mithilfe der bereits genannten `XMLHttpRequest` bzw. des `ActiveXObjects` ans IVS gesendet werden. Die Antwort des IVS wird dann entsprechend weiterverarbeitet und beispielsweise in Form eines Kamerabildes direkt ausgegeben.

2. Integration von Jira und Confluence

Das nächste größere Projekt war die Integration der Atlassian-Produkte *Jira*[4] und *Confluence*[5] in den Firmenalltag. Diese beiden Programme sind für das Production Management gedacht und haben das Ziel, einen durchgängigen, geplanten und leicht nachvollziehbaren Arbeitsfluss zu erreichen.

Confluence ist dabei ein einfaches System, dessen Ziel die einfache Weitergabe von Informationen ist. Es dient der Erstellung von Dokumenten, welche externe Daten wie beispielsweise Microsoft-Office-Dokumente oder YouTube-Videos sowie Bilder usw. enthalten können und vereinfacht die Zusammenarbeit mehrerer Personen an einem Dokument.

Jira arbeitet im Zusammenspiel mit Confluence und ist ursprünglich für die Softwareentwicklung gedacht. Entwicklungsvorgehensweisen wie *Scrum* o.Ä. können hier ohne Umwege umgesetzt werden und Jira bietet Funktionalitäten wie Sprints oder Burndown-Charts an, um den Arbeitsfluss zu sichern.

Die Verwendung von Jira und Confluence sollte das davor eingesetzte (und veraltete) Tool *easyPM* ersetzen, welches für die Planung von Aufgaben, Abläufen und Verwaltung von Zeitspalten eingesetzt wurde. Informationen wie E-Mails, Anrufprotokolle oder sonstige Daten waren hier jedoch dezentral, bzw. konnten nicht direkt in easyPM gespeichert werden. Durch das Zusammenspiel der beiden neuen Systeme sollte dieses Problem behoben und die Speicherung der Daten konsistent werden.

[4] https://de.wikipedia.org/wiki/Jira_(Software)
[5] https://de.wikipedia.org/wiki/Confluence_(Atlassian)

2.1 Einrichtung der Datenbank

Jira und Confluence sind beides Tools, die über einen Server laufen und dann über den Webbrowser der Endgeräte aufgerufen werden können. Dies sichert zum einen die Zentralität der Datenspeicherung und -verarbeitung und entlastet andererseits die Nutzergeräte stark. Weiterhin ist die Nutzung eines firmeninternen Servers (welcher außerdem für andere Zwecke wie das Intranet schon vorhanden war) sinnvoll, da somit die Sicherheit der gespeicherten Daten gewährleistet wird.

Für den Aufbau des Webservers wird dabei sowohl von Jira als auch von Confluence eine Datenbank verwendet. Das kann sowohl eine *MySQL*-, *Oracle*- oder *Microsoft-SQL*-Datenbank sein, oder aber auch die Open-Source-Alternative *PostgreSQL*[6]. Wegen der eher schlechten Unterstützung von MySQL habe ich schlussendlich eine PostgreSQL-Datenbank eingesetzt.

PostgreSQL ist ein freies und quelloffenes Datenbankmanagementsystem. Es wird von Betriebssystemen wie Windows, Linux und anderen UNIX-ähnlichen Systemen unterstützt und wird bei den meisten Linux-Distributionen bereits mitgeliefert.

Um den Linux-Server von isys zu simulieren und die Datenbank entsprechend einzurichten, habe ich eine virtuelle Maschine mit einer Ubuntu-Installation verwendet. Nach der Einrichtung des Systems konnte ich mit der Konfiguration des Datenbankservers beginnen. Mit einigen einfachen SQL-Befehlen waren die notwendigen Datenbanken dann auch schnell erstellt:

```
1 CREATE DATABASE confluencedb WITH ENCODING 'UNICODE';
2 GRANT ALL PRIVILEGES ON DATABASE confluencedb TO confluence;
3
4 CREATE DATABASE jiradb WITH ENCODING 'UNICODE';
5 GRANT ALL PRIVILEGES ON DATABASE jiradb TO jira;
```

Listing 2.1: SQL-Befehle zur Erstellung der Datenbanken

Zur Anbindung an Jira bzw. Confluence musste nun nur noch bei der Einrichtung jeweils die entsprechenden Datenbankdaten angegeben werden. Da hier der Jira- bzw. Confluence-Server und die Datenbank auf dem selben Rechner laufen, ist der Host beispielsweise `127.0.0.1` bzw. `localhost`.

[6] `https://de.wikipedia.org/wiki/PostgreSQL`

Die Installation von Jira und Confluence erfolgte dann einfach über die Ausführung eines Shell-Skriptes und die Eingabe einer Lizenz. Somit war die Einrichtung der Systeme vollständig und ich konnte einige Einstellungen vornehmen, z.B. Anpassungen im Design der Seiten, sodass die Oberfläche den Farben der Firma entsprach (Grün [#006833] und Rot [#9B2536]).

2.2 Einrichtung des Intranetservers

Nun, da Jira und Confluence online und konfiguriert waren, setzte ich mich mit der Einrichtung eines Servers auseinander, welcher im Intranet betrieben werden und die beiden Webtools hosten sollte.

Im ersten Schritt machte ich den dafür zur Verfügung gestellten Rechner nutzbar, indem ich eine frische Linux-Installation vornahm. Auf dieser Basis konnte ich nun mit `VirtualBox`[7] eine virtuelle Maschine einrichten; auch darauf lief ein Linux-Betriebssystem. Die Nutzung einer virtuellen Maschine hatte hierbei mehrere Vorteile:

1. **Flexibilität**: Durch das Speichern in einem **Disk Image**[8] ist der Ort des Servers sehr leicht austauschbar. Durch Verschieben des Images sind sämtliche Daten des Servers übernommen, was die Wartung stark vereinfacht.

2. **Sicherheit**: Da die Daten in einem einzigen Image liegen, welches auf der Festplatte des Wirtrechners gespeichert wird, ist die Datensicherung bzw. Erstellung von Backups besonders komfortabel. Mithilfe eines RAID-Systems wird in diesem Fall die Datensicherheit gewährleistet und Backups erstellt.

3. **Vielseitigkeit**: Der Server läuft in einer virtuellen Maschine, welche auf die Ressourcen des Wirtrechners zugreift. Durch das Einrichten einer weiteren virtuellen Maschine kann dessen Rechenleistung und Speicherkapazität optimal ausgenutzt werden und die Maschine somit auch für andere Zwecke verwendet werden.

Nach erneuter Einrichtung der in Kapitel 2.1 beschriebenen Datenbank und der Anbindung an Jira/Confluence musste ich mich nun noch mit der Netzwerkverbindung auseinandersetzen. Die zugeordnete IP-Adresse sollte dabei `10.0.0.14` sein; Jira und Confluence jeweils unter den

[7] `https://wiki.ubuntuusers.de/VirtualBox/Installation/#VirtualBox-OSE-Open-Source-Edition`

[8] `https://de.wikipedia.org/wiki/Speicherabbild`

Standardports `8080` bzw. `8090` erreichbar sein.

Als Erstes musste die virtuelle Maschine über eine Netzwerkbrücke auf das lokale Netzwerk zugreifen können. Dies war schnell eingestellt; eine einfache Änderung im VirtualBox-Interface genügte dafür. Mithilfe der Netzwerkbrücke wird hier das lokale Netzwerk (isys vision Intranet) mit dem von der virtuellen Maschine erstellten Netzwerk verbunden, sodass diese die gehosteten Services auf die Webports anlegen kann.

Im nächsten Schritt braucht die virtuelle Maschine eine feste (statische) IP-Adresse. In diesem Fall sollte dies die `10.0.0.14` sein. Um diese zuzuweisen, kann unter Linux die Datei `/etc/network/interfaces` bearbeitet werden:

```
1 auto enp0s3 #Name der Netzwerkbrücke
2 iface enp0s3 inet static #IP auf Statisch ändern
3 #Setzen von Verbindungsinformationen
4 netmask 255.255.0.0
5 gateway 10.0.0.10
6 network 10.0.0.0
7 broadcast 10.0.0.255
8 dns-nameservers 10.0.0.10
```

Listing 2.2: Änderungen in `/etc/network/interfaces`

Mithilfe dieser Änderungen wird dem Rechner eine statische Adresse zugewiesen. Dies ist hier notwendig, da der Rechner sonst bei jedem Start vom Router eine neue, variable Adresse zugewiesen bekommt. Damit der Server also immer mit der selben Adresse erreichbar ist, muss die IP fest sein. Später lässt sich auch eine DNS zuweisen, beispielsweise `intranet.isys-vision.de/jira`.

2.3 Ausfallsicherheit über ein RAID-System

Um die Stabilität des Servers im Intranets zu gewährleisten, sollte das System durch ein RAID-5[9] abgesichert werden. Ein RAID-5 besteht aus mindestens 3 unabhängigen Festplatten, welche dieselben Daten redundant abspeichern. Beim Ausfall einer der Platten wird somit sichergestellt, dass die Daten weiterhin verfügbar (und auch nutzbar) bleiben.

[9] `https://de.wikipedia.org/wiki/RAID#RAID_5`

Da Hardware-RAIDs oft sehr teuer sind, habe ich auf ein Software-RAID zurückgegriffen. Hierbei wird die Redundanz der Daten durch Software erreicht, welche das Plattenmanagement übernimmt. Das Betriebssystem verblieb dabei auf der vorherigen Platte, sodass nur die virtuelle Maschine auf dem RAID gespeichert wird.

Unter Linux lässt sich ein RAID mit dem Tool `mdadm` sehr gut realisieren. Man kann dabei auswählen, welches RAID-Level gewählt (in diesem Fall ein RAID-5-System) und welche und wie viele Festplatten genutzt und werden sollen.

3. Web-Tools für das Mikado-Projekt

Nach meiner Arbeit am IVS, Jira und Confluence wurde ich nun Teil eines anderen Projekts namens Mikado[10]. Mikado ist ein Robotikprojekt, bei dem es hauptsächlich um das sogenannte "Bin Picking" geht. Dabei wird ein Roboter dazu eingesetzt, zufällig in einer Kiste liegende Teile mithilfe einer 3D-Kamera zu erkennen, zu greifen und in eine andere Position zu bringen. Die Herausforderung liegt hierbei beim Leeren der Kiste, d.h. alle sich darin befindlichen Teile gleichgültig ihrer Position greifen zu können.

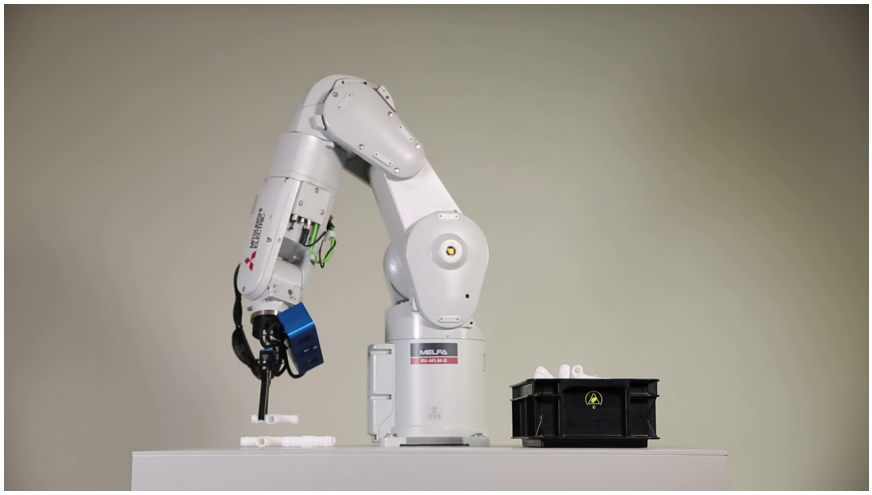

Abbildung 3.1: Mikado-Roboter von Mitsubishi[11]

3.1 Demoablauf für Messen

Zum Kennenlernen der Programmschnittstelle und des Roboters sollte ich zunächst mit dem von isys geschriebenen Programm einen Demoablauf aufbauen, welcher für Messen oder ähnliche Anwendungsfälle verwendet werden sollte. Im Programm lassen sich dafür Abläufe graphisch regeln, beispielsweise eine Bewegung, das Greifen eines Teils oder die Bildaufnahme mit der 3D-

[10] http://www.mikado-robotics.de
[11] Quelle: https://img.youtube.com/vi/L7nppfZ6pKg/maxresdefault.jpg

Kamera. Diese noch nicht veröffentlichte Software konnte somit außerdem von mir getestet und Verbesserungsvorschläge geäußert werden.

Zur Erkennung der Teile wird mithilfe der Kameras ein 3D-Bild aufgenommen, welches anschließend von der Recheneinheit analysiert und zu einer Punktwolke transformiert wird. Diese Punktwolke wird nun mit dem eintrainierten CAD-Modell abgeglichen und somit mögliche Kandidaten anhand ihrer Übereinstimmung identifiziert. Die dabei gewählte Mindestübereinstimmung lässt sich dabei einstellen, was je nach Wert Vor- bzw. Nachteile mit sich bringt:

Höhere Mindestqualität	**Niedrigere Mindestqualität**
Erkannte Teile können mit höherer Sicherheit richtig und sicher gegriffen werden	Erkannte Teile können sich in nahezu ungreifbarem Winkel befinden und daher nicht sicher gegriffen werden
Weniger Teile werden erkannt, daher höhere Chance, die Kiste nicht leer zu bekommen	Viele Teile werden erkannt, die Wahrscheinlichkeit ist daher größer, alle verbleibenden Teile zu identifizieren

Tabelle 3.1: Vor- bzw. Nachteile einer höheren vs. niedrigeren Mindestqualität

Die Bewegungen des Robotors lassen sich entweder über die Winkel der einzelnen Gelenke (beim von isys verwendeten Roboter von Mitsubishi Electrics sind es 6 Gelenke) oder das Format `xyz rpy` (`xyz` ist hierbei die Position im Raum (3 Koordinaten), `rpy` (roll, pitch, yaw) die Drehung des Greifers) angeben. Sämtliche Bewegungen und Rotationen sind dabei kollisionsgeprüft, d.h. sobald das System eine mögliche Kollision erkennt, wird die Bewegung nicht ausgeführt. In diesem Fall kann eine alternative Aktion durch einen Sprung im Programmablauf durchgeführt werden.

Über einzelne Sektionen und Sprünge lässt sich nun ein strukturiertes Programm erstellen, was sowohl einfache Funktionen (wie in diesem Fall eine Demonstration der Funktionalität) oder auch komplexe Abläufe wie den Zusammenbau eines Schneckengewindes übernehmen kann. Der Programmablauf kann dabei mit den Funktionen `start`, `pause`, `step` und `reset` gesteuert werden, was in folgenden Aufgaben noch wichtig wird.

Mein Demonstrationsprogramm war hierbei die Abwandlung einer älteren Demo. Der Roboter nimmt dabei Zahnräder aus einer Kiste, testet mithilfe einer Lichtschranke, ob er das Teil tat-

sächlich gegriffen hat, und legt dieses schlussendlich so ab, dass der Schriftzug "isys" entsteht. Ist dies geschehen, nimmt er nacheinander die abgelegten Teile wieder auf und legt sie zurück in die Kiste.

Der Zweck der Demonstration ist einerseits das Heranführen an den Roboter an sich (z.B. die Arbeitsweise der Maschine oder die graphische Programmierung) sowie die Fähigkeit, wirklich sämtliche Teile aus der Kiste nehmen zu können, zu demonstrieren. Erreichen lässt sich dies einfach dadurch, dass zu Beginn nur so viele Zahnräder in die Kiste gelegt werden, wie am Ende auch für den Schriftzug benötigt werden.

3.2 Web-Remote für die Mikado-Software

Für die Demonstration bei Messen oder bei ähnlichen Veranstaltungen sollte nun von mir ein Webinterface erstellt werden, was als Fernzugriff für die Mikadosoftware verwendet werden kann. Der Sinn dahinter ist, dass der Programmablauf nicht mehr ausschließlich mithilfe von Bildschirm und Tastatur gesteuert, sondern beispielsweise über ein Smartphone modern und komfortabel bedient werden soll. Dies ist insbesondere nützlich, da aus Sicherheitsgründen bei Vorführungen eine Glaswand vor dem Roboteraufbau ist, was Eingriffe in den Ablauf stark erschwert.

3.2.1 Design der neuen Webseite

Nachdem ich im 1. Projekt bereits mit Javascript, CSS und HTML in Berührung gekommen war, konnte ich nun eine völlig neue Webseite basierend auf meinen erlangten Kenntnissen selbstständig und von Grund auf aufbauen. Dazu kümmerte ich mich zuerst um Inhalt und Design der neuen Seite und fügte anschließend die Funktionalität durch Javascript und jQuery hinzu.

Struktur mit HTML

Zunächst gliederte ich die neue Webseite mithilfe der zugehörigen HTML-Tags in sinnvolle Abschnitte. Mit `<nav>`, `<main>` und `<footer>` sowie geschachtelten `<div>`-Tags konnte ich eine sinnvolle Struktur in den Inhalt der Seite bringen.

Im `<nav>`-Teil der Seite wird die Navigationsleiste definiert; in meinem Fall umfasste sie die

Listenpunkte **Logs**, **Kontrolle** und **Über** sowie ein **Hamburger-Menü-Icon**[12], welches zum Ein- und Ausklappen des Menüs notwendig war. Im `<main>`-Teil des HTML-Dokuments wird sämtlicher Inhalt der Seite definiert, also in meinem Fall das Logfenster, die Kontroll- und die Informationsseite. Diese einzelnen Abschnitte sind jeweils wieder in einzelne `<div>`s geteilt, sodass Struktur und Design gewährleistet werden können. Schlussendlich befinden sich im `<footer>`-Element einige Informationen über das Produkt, wie z.B. Copyright, Webseite oder Kontakt.

```
<div id="control" class="content">
  <div class="control-window">
    <div class="control-element">
      <button onclick="start()">
        <img src="media/start.svg" alt="Start">
      </button>
    </div>
    <div class="control-element">
      <button onclick="step()">
        <img src="media/step.svg" alt="Step">
      </button>
    </div>
    <div class="control-element">
      <button onclick="pause()">
        <img src="media/pause.svg" alt="Pause">
      </button>
    </div>
    <div class="control-element">
      <button onclick="reset()">
        <img src="media/reset.svg" alt="Reset">
      </button>
    </div>
  </div>
</div>
```

Listing 3.1: Struktur der Kontrollseite

Wie man sieht, bestehen die Kontrollbuttons aus einem Funktionsaufruf (den ich erst später

[12] `https://de.wikipedia.org/wiki/Hamburger-Men%C3%BC-Icon`

implementiert habe) und einer SVG-Grafik[13].

Erstellen von SVG-Grafiken

SVG-Grafiken haben den großen Vorteil, dass sie vom Browser gerendert werden. Dadurch ist einerseits keine große Bilddatei notwendig, was (insbesondere auf mobilen Endgeräten) zu sehr schnellen Ladezeiten führt, und andererseits auch nicht das Problem zu geringer Auflösungen mit sich bringt, wodurch die Grafik immer gestochen scharf ist. Da jedoch keine entsprechenden Grafiken vorhanden waren, musste ich diese selbst erstellen. Auch für das Navigationsmenü habe ich SVG-Dateien entworfen.

Der Inhalt einer SVG-Grafik ist der Syntax von HTML bzw. XML sehr ähnlich. Zunächst wird mit dem Tag <svg> gekennzeichnet, dass es sich um eine SVG-Grafik handelt. Innerhalb dieses Tags können nun Bildelemente hinzugefügt werden:

```
1  /<svg width="54" height="54" viewBox="0␣0␣54␣54"
       xmlns="http://www.w3.org/2000/svg" xmlns:xlink=
       "http://www.w3.org/1999/xlink">
2    <circle
3      cx="27" cy="27" r="25"
4      stroke="black" stroke-width="4" fill="none"
5    />
6    <line
7      x1="18" y1="13"
8      x2="18" y2="41"
9      stroke="black" stroke-width="8" stroke-linecap="butt"
10   />
11   <line
12     x1="36" y1="13"
13     x2="36" y2="41"
14     stroke="black" stroke-width="8" stroke-linecap="butt"
15   />
16 </svg>
```

Listing 3.2: Bildelemente in einer SVG-Grafik

[13] https://de.wikipedia.org/wiki/Scalable_Vector_Graphics

Hier wird beispielsweise zunächst ein umschließender Kreis definiert (ll. 2 - 5), welcher den Mittelpunkt $P(27|27)$ und den Radius $r = 25$ hat. Die Linienfarbe soll Schwarz sein, die Linienbreite 4 Pixel und der Kreis soll nicht gefüllt sein. Anschließend werden zwei Linien gezeichnet, jeweils mit der Farbe Schwarz, der Breite 8px und einem geraden Linienende. Linie 1 geht in diesem Fall von $P(18|13)$ nach $Q(18|41)$.

Stilregeln durch CSS

Im nächsten Schritt ging es um das Design mithilfe von CSS. Die vorhergehend definierten Inhalte konnte ich nun schöner Darstellen, indem ich Stilregeln für bestimmte Elemente und Klassen vornahm. Dazu gehörten unter Anderem die Schriftart, Farbschema für Menüleiste und Links oder auch Positionierung von Abschnitten und Festlegen von Abständen.

Sehr wichtig war hierbei die Orientierung an relativen Werten wie der Bildschirmweite (`viewport width`). Da es sich hierbei um ein für Mobilgeräte optimiertes Webinterface handelt, spielt Skalierbarkeit auf unterschiedliche Größen eine große Rolle. Umsetzen lässt sich dies beispielsweise durch ein sogenanntes **Responsive Design**[14], bei dem die Webseite der Bildschirmgröße entsprechend reagiert. Dies sichert auch eine gute und intuitive Darstellung beim Wechsel vom Portrait- in den Landschaftsmodus (z.B. bei Tablet PCs).

In der Umsetzung erfolgt dies durch die Verwendung von relativen Werten (wie beispielsweise `width: 1em;`) und `@media`-Tags, welche für die Skalierbarkeit der Webseite und den enthaltenen Elementen verantwortlich sind.

[14] `https://de.wikipedia.org/wiki/Responsive_Webdesign`

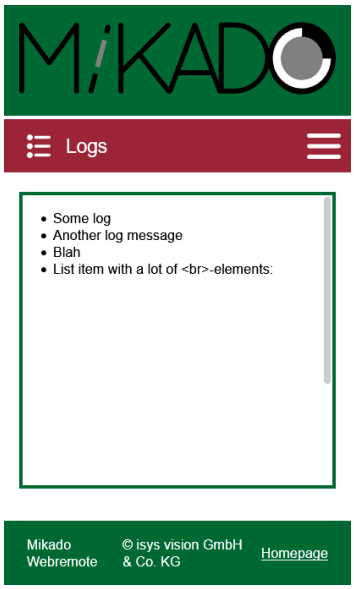

Abbildung 3.2: Logfenster

Abbildung 3.3: Kontrolle

Abbildung 3.4: Info / Menü

3.2.2 Befehls- und Logübertragung mithilfe von RosJS

Auf dem Hostrechner des Roboters läuft zur Kontrolle von diesem eine Python-Bibliothek namens ROS (Roboter Operating System). Sie bietet viele Funktionalitäten, beispielsweise zum Steuern der einzelnen Gelenkwinkel durch Python.

Weiterhin werden Services zur Ausführung von einzelnen Aufgaben bereitgestellt. Der Service `/reset` setzt beispielsweise den aktuellen Programmablauf zurück und kann mit `rosservice call /reset` aufgerufen werden. Es sind außerdem die Services `/run` und `/pause` vorhanden, welche die für mich notwendige Funktionalität beinhalten. Die beiden letzteren werden mit Parametern aufgerufen: `rosservice call /run "step:false blocking:false"`, bzw. `rosservice call /pause "blocking:false"`.

Der Parameter `blocking` ist dabei in meinem Fall irrelevant; umso interessanter ist jedoch das Argument `step`: Ist es auf `false` gesetzt, läuft das Programm beim Aufruf normal durch. Sobald der Service jedoch mit `step:true` aufgerufen wird, wird nur ein Schritt im Programmablauf durchgeführt.

Zur Anbindung an das von mir entwickelte Webinterface habe ich nun die Bibliothek RoslibJS verwendet, welche als Kommunikationsmodul für ROS fungiert. Hiermit können entsprechende Aufrufe im Javascript festgelegt werden, um diese anschließend an den Hostrechner zu senden und somit den zugehörigen Service zu starten.

Meinen <button>-Elementen hatte ich jeweils die zugehörige Funktion im Argument onclick zugeordnet (Beispiel: <button onclick="pause()">Pause</button>). Im zugehörigen Javascript sind nun einmal die generelle Einrichtung von RoslibJS (wie z.B. Verbindungsinformationen etc.) und für jeden Button die zugehörige Funktion vorhanden.

```
1  var ros = new ROSLIB.Ros();
2  ros.on('error', function(error) {/* Log weggelassen */});
3  ros.on('connection', function() {});
4  ros.on('close', function(){});
5  ros.connect('ws://10.0.4.136:8080');
6  /* Irrelevante Funktionen weggelassen */
7  function start() {
8    var rosStart = new ROSLIB.Service({
9      ros: ros, name: '/run', messageType: 'mikado_msgs/Run',
10   });
11   var rosStartRequest = new ROSLIB.ServiceRequest({
12     step: false, blocking: false,
13   });
14   rosStart.callService(rosStartRequest);
15 }
16 /* Funktionsrumpf weggelassen */
17 function step() {};
18 function pause() {};
19 function reset() {};
20 /* Funktion zum Anhängen von Logs an das Logfenster. */
21 function initLog() {
22   var rosLog = new ROSLIB.Topic({
23     ros: ros, name: '/rosout', messageType: 'rosgraph_msgs/Log',
24   });
25   rosLog.subscribe(function(message) {
26     var log = '<li␣class="';
```

```
27      switch (message.level) {
28        case 1:
29          log += "debug";
30          break;
31      } //Other cases omitted
32      log += '">' + message.msg + '</li>';
33      addLog(log);
34    });
35  }
```

Listing 3.3: Serviceaufruf durch RoslibJS

Zunächst wird in den ersten 5 Zeilen die Verbindung zum ROS hergestellt. In den anschließend folgenden Funktionen wird zunächst ein neuer Service für die aktuelle Routine erstellt (Zeilen 8 - 12), welcher dann mit einer Anfrage (Zeilen 13 - 16) gestartet wird (Zeile 17). Der Service selbst enthält hierbei Informationen über den aufgerufenen Befehl (in diesem Fall `/run`), während die Anfrage Informationen über die Parameter enthält (hier die Argumente `step` und `blocking`).

In der Funktion `initLog()` in Zeile 24 wird eine Verbindung zum ROS hergestellt, um neue Lognachrichten ans Logfenster anzuhängen. Zunächst wird dafür ein sogenanntes "Topic" erstellt, welches dann mithilfe von `subscribe()` laufend aktualisiert werden kann. Den hier ankommenden Nachrichten wird nun mithilfe des Wichtigkeitslevels eine entsprechende CSS-Klasse zugewiesen (z.B. haben Meldungen vom Typ `debug` die Farbe grün, `error` ist rot usw). Anschließend werden die eigentlichen Logmeldungen an die bereits existierende Liste angehängt und erscheinen somit im Logfenster.

3.2.3 Webhosting und Netzwerk

Nachdem das Interface nun eigentlich die gewünschte Funktionalität besitzt, muss es noch erreichbar gemacht werden. Dazu wird ein HTTP-Server verwendet, welcher die Webseite hostet. Mit der Python-Bibliothek `flask` ist dies recht einfach zu realisieren:

```
1 import rospy
2 from flask import Flask, render_template
3 app = Flask(__name__)
4 @app.route('/')
```

```
5 def render_static():
6     return render_template('index.html')
7 if __name__ == '__main__':
8     app.run()
```

Listing 3.4: Python-Skript zum Hosten des Webservers

Nun lässt sich unter der IP-Adresse des Rechners (lokal `127.0.0.1`) die gehostete Webseite aufrufen. Über ein WLAN-Netzwerk lässt sich nun eine Verbindung zu mobilen Endgeräten aufbauen, welche dann auf das Webinterface zugreifen und somit den Roboter kontrollieren können.

3.3 Web-Preview zur Darstellung von Punktwolken

Zur Objekterkennung beim Mikado-Roboter wird von diesem mithilfe der 3D-Kamera eine sogenannte Punktewolke aufgenommen - eine Ansammlung von Punkten, welche jeweils mit ihren Koordinaten abgespeichert werden. Somit wird eine Art "3D-Gitter" erzeugt, welches dann mit dem vogegebenen Modell verglichen werden kann.

Zur Veranschaulichung sollten diese Punktewolken nun mithilfe von Javascript in einer einfachen Seite visualisiert werden. Zuvor wurde auf umständliche und veraltete Methoden zurückgegriffen, beispielsweise wurde ein Video aufgezeichnet und per Mail verschickt. Die Nutzung eines Javascript-Frameworks ist hierbei sehr sinnvoll und komfortabel, da

1. Daten sehr flexibel und schnell zur Nutzung im Webinterface exportiert werden können
2. durch die Darstellung im Browser nahezu jedes Endgerät die Möglichkeit hat, die Webansicht zu nutzen
3. das Framework auf einem Server von isys gehostet werden kann und dann für jeden Kunden eine unterschiedliche Simulation geladen werden kann
4. keine Installation auf Seite der Kunden notwendig ist.

3.3.1 Rendering mit `THREE.js`

Zur Darstellung der Punktewolke habe ich `THREE.js`[15] verwendet. Dabei handelt es sich um ein Framework, welches auf OpenGL bzw. WebGL basiert und Funktionalitäten zum Rendering von Objekten bereitstellt.

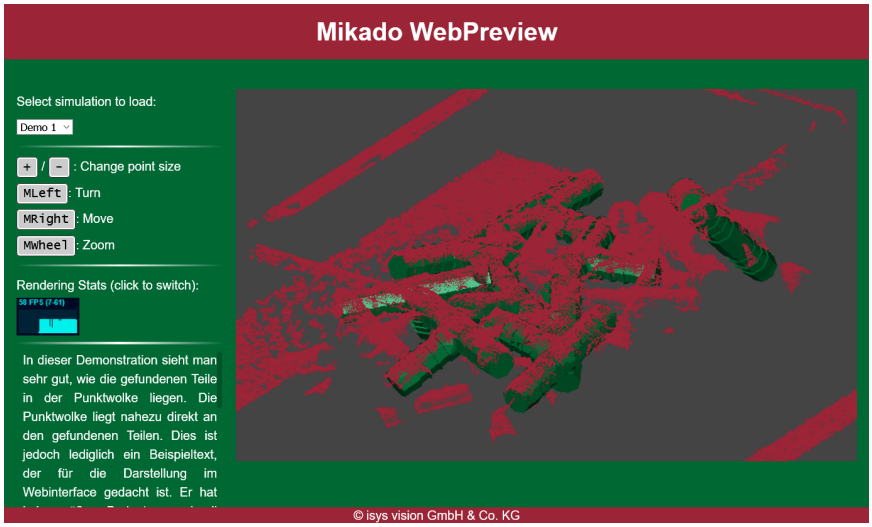

Abbildung 3.5: Punktwolke mit Beispielfunden in der Webvorschau

Nachdem ein Grundgerüst aufgebaut wurde, welches THREE.js einbindet, kann der Renderer nun initialisiert werden. Das Framework enthält dabei viele Methoden und Objekte, welche nun entsprechend deklariert und genutzt werden können. Da es sich um viele kleine Konfigurationsschritte handelt, ist folgendes Listing stark gekürzt.

```
1  fetch('index.json');
2  function loadSTL(model, pose) { //Start loading found meshes
3    fetch(pose); //Load position from .json file
4    for (/*...*/) {
5      scene.add(mesh); //Add meshes of found items
6    }
7  }
8  function loadPCD(filename) {
9    var pcdLoader = new THREE.PCDLoader(); //Load PCD file
10   pcdLoader.load(filename /* material, position etc. omitted */);
```

[15] https://threejs.org/

```
11 }
12 function load() {
13   /* create objects like renderer, scene, camera... */
14 }
15 function animate() {
16   requestAnimationFrame(animate);
17   renderer.render(scene, camera);
18   controls.update();
19 }
20 function resizePoints(factor) {
21   var pointCloud = scene.getObjectByName('pointcloud');
22   pointCloud.material.size *= factor;
23   pointCloud.material.needsUpdate = true;
24 }
```

Listing 3.5: Initialisierung von THREE.js

Zunächst wird in Zeile 1 die Datei `index.json` geladen. Sie enthält wichtige Informationen wie eine Liste aus Punktwolken-Fundposition-Zusammenhängen sowie eine Beschreibung der Szene. Nachdem sie geladen wurde, werden Funktionen wie `load()` aufgerufen, die Simulation gestartet usw.

Die Funktionen `loadSTL` und `loadPCD` laden die `.stl`-Dateien der Funde bzw. die Punktwolke und setzen die in der JSON-Datei festgelegte Position der Teile. Sie werden bei Auswahl einer anderen Demo im Interface aufgerufen.

Die Funktion `load()` wird nun zur Initialisierung der Simulation aufgerufen und erstellt einen Renderer, eine Szene (in der das Bild dargestellt wird) und die Kamera.

Die Funktion `animate()` ist eine einfache Update-Funktion, welche das angezeigte Bild aktualisiert und mit jedem Frame aufgerufen wird. Hier wird die Tastatureingabe gelesen bzw. auf einen relevanten Tastendruck gewartet.

In `resizePoints` wird die Größe der Punkte der Punktwolke geändert. Sie wird beim Tastendruck oder Klick auf einen Button im Interface aufgerufen.

3.3.2 Konfiguration durch JSON-Dateien

Das durch Bibliotheken und eigenes Javascript nun fertiggestellte Webtool sollte nun, je nach Kunde, mit unterschiedlichen Funddaten genutzt werden. In der Simulation sollte die Auswahl von festgelegten Demonstrationen durch ein Dropdown-Menü ermöglicht werden. Dafür wird in der Datei `index.json` eine Konfiguration erstellt, welche dann vom Javascript geladen, eingelesen und verwendet wird. Diese Datei enthält eine Liste von Demonstrationen. Sie besteht also aus mehreren Einträgen, welche dann als Optionen im Dropdown-Menü angezeigt werden.

```
1 [ {
2   "pcd" : "pcdFile.pcd",
3   "pcdColor" : "0x9b2436",
4   "stl" : "stlFile.stl",
5   "stlColor" : "0x006833",
6   "pose" : "pose.json",
7   "description" : ""
8 }, {} ]
```

Listing 3.6: Inhalt von `index.json`

Eine solche Option beinhaltet sechs Eigenschaften, deren Werte für die Darstellung verantwortlich sind:

- pcd: Dateipfad zur Punktwolkendatei, die geladen wird.
- pcdColor: Die Farbe, in der die Punktwolke dargestellt wird.
- stl & stlColor: analog zu den ersten beiden Eigenschaften.
- pose: Pfad zur JSON-Datei, welche Informationen über die Funde beinhaltet.
- description: Eine Beschreibung, die zusätzlich zur Animation angezeigt werden soll.

Die mit dem Namen "pose" gekennzeichnete Datei gibt dabei an, wie die STL-Modelle in der Punktwolke ausgerichtet werden. Sie beinhaltet sowohl Koordinaten der einzelnen Funde als auch die Rotation im Axis-Angle-Format[16]. Diese wird, wie auch die Punktwolke und Beschreibung, von isys-Mitarbeitern angelegt, und kontrolliert damit die Simulation. Die Informationen über das Fundstück, also die STL-Datei, stammen jedoch vom Kunden.

[16] `https://en.wikipedia.org/wiki/Axis-angle_representation`

4. Persönliche Bilanz und Ausblick

Während meiner Tätigkeit habe ich vielseitige Aufgaben und Probleme gelöst und konnte so in fast alle Unternehmensbereiche einen Einblick bekommen. Einerseits konnte ich mit den Entwicklern des IVS zusammen arbeiten und so einen Blick auf die teilweise noch sehr alte Technologie werfen. Im starken Gegensatz dazu war ich auch beim noch sehr jungen Mikado-Projekt aktiv und konnte damit die zukunftsorientierte Arbeitsweise des Unternehmens erfahren. Außerdem war ich bei der Einrichtung des Jira-Servers vollständig autonom, d.h. an keinem größeren Projekt direkt beteiligt. Hier habe ich die Technik im Production Management, Continuos Integration und der Versionsverwaltung kennen gelernt.

Mit diesen Einblicken konnte ich mich sowohl fachspezifisch, fächerübergreifend als auch persönlich weiterbilden und viele wertvolle Erfahrungen sammeln. Besonders meine Fachkenntnisse zu Webentwicklung, insbesondere Javascript, konnte ich vertiefen. Außerdem wurde ich auch mit den Problemen der Entwicklung in Teams konfrontiert, beispielsweise der Wichtigkeit von korrekt dokumentiertem Code. Weiterhin erlangte ich einen Blick auf den Firmenalltag und Abläufe, was für mich neu war.

Die während meines Studiums erlangten Kenntnisse zu Datenbanken, Algorithmik und Programmiertechnik haben mir dabei stark weitergeholfen und mich gut auf die Arbeit im Unternehmen vorbereitet.

Besonders das weitreichende Aufgabenfeld und die freundschaftliche Beziehungen zu den Kollegen haben das Praktikum positiv geprägt. Mein einziger Kritikpunkt ist die lange Wartezeit, die ich zwischen verschiedenen Aufgaben hatte. Dies ist vermutlich jedoch auf die geringe Erfahrung der Firma mit studentischen Praktikanten zurückzuführen. Generell kann ich ein Praxissemester bei isys vision aber absolut empfehlen.

Ehrenwörtliche Erklärung

Hiermit erkläre ich, dass ich den vorliegenden Bericht selbstständig angefertigt und alle von mir genutzten Quellen und Hilfsmittel angegeben habe. Dieser Bericht wurde bisher in keiner anderen Prüfungsbehörde oder Person im Rahmen einer Prüfung vorgelegt und auch nicht veröffentlicht.

Datum

Unterschrift