

**Fachrichtung Informatik**

**Schuljahr 2019/2020**

# **DIPLOMARBEIT**

Gesamtprojekt

# **Digitaler Zwilling Ultraschallprüfanlage mit AR**

**Ausgeführt von:**

Deubler Jakob, 5BHIF-06

Dimitrov Aleks, 5BHIF-07

Ecker Christof, 5BHIF-08

**Betreuer:**

DI Alfred Doppler

Ried im Innkreis, am 03.04.2020

---

**Abgabevermerk:**

**Datum:**

Betreuer: DI Alfred Doppler

## **Erklärung gemäß Prüfungsordnung**

„Ich erkläre an Eides statt, dass ich die vorliegende Diplomarbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und alle den benutzten Quellen wörtlich oder sinngemäß entnommenen Stellen als solche kenntlich gemacht habe.“

Ried im Innkreis, 31.03.2020

Verfasser:


Jakob Deubler

Aleks Dimitrov

Christof Ecker

## **Gender Erklärung**

Aus Gründen der Lesbarkeit wird innerhalb dieser Diplomarbeit darauf verzichtet, geschlechtsspezifische Formulierungen zu verwenden. Soweit personenbezogene Bezeichnungen nur in männlicher Form angeführt sind, beziehen sie sich auf Männer und Frauen in gleicher Weise.


	<b>HTBLA Grieskirchen</b>	
	Fachrichtung:	<b>Informatik</b>

## DIPLOMARBEIT

### DOKUMENTATION


<b>Namen der Verfasser/innen</b>	Jakob Deubler  Aleks Dimitrov  Christof Ecker
<b>Jahrgang Schuljahr</b>	5BHIF 2019/2020
<b>Thema der Diplomarbeit</b>	Digitaler Zwilling Ultraschallprüfanlage mit AR
<b>Kooperationspartner</b>	Fill GmbH

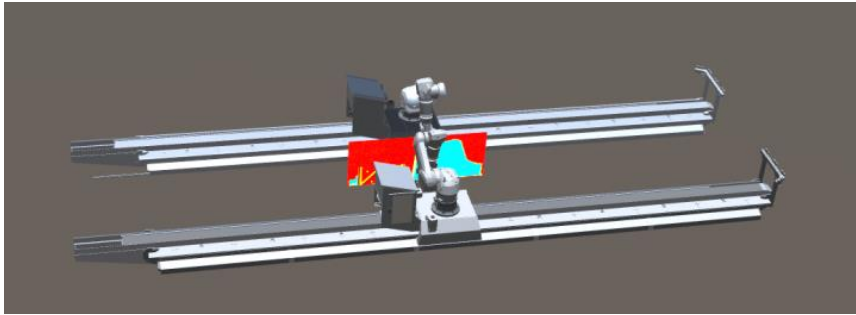
<b>Aufgabenstellung</b>	<p>Das fertige Programm soll den UltrasonicInseptionDualRobot besser veranschaulichbar machen, welcher ein Bauteil auf Materialfehler überprüft. Die Messergebnisse sollen parallel in Form einer Heatmap auf dem Bauteil abgebildet werden. In Zukunft will die Firma Fill auf Messen oder ähnlichen Veranstaltungen den Kunden die Funktionsweise ihrer Anlage mit einer simplen AR-Projektion zeigen.</p>
-------------------------	--

	<b>HTBLA Grieskirchen</b>	
	Fachrichtung:	<b>Informatik</b>

<b>Realisierung</b>	<p>Die Modelle (Roboter, Bauteile, Werkzeuge) wurden in Blender realisiert. Die Funktionalität wurde in der Entwicklungsumgebung Unity umgesetzt. Für die Augmented Reality Projektion wurde die Vuforia-Bibliothek appliziert.</p> <p>Die Versionsverwaltung des Projekts wurde mit Hilfe von Unity-Collabrate verwirklicht. Zusätzlich nutzten wir für anderweitigen Datenaustausch Google Drive. Die Kommunikation untereinander und mit dem Auftraggeber fand auf Telegram statt.</p>
---------------------	---

<b>Ergebnisse</b>	<p>Das Ergebnis der Diplomarbeit besteht aus dem Windows-Programm und der Android-Apk.</p> <p>Mit dem Windows-Programm kann die Funktionsweise der Ultraschallprüfanlage simple dargestellt werden. Zusätzlich können die digitalen Roboter, von der Steuerung der echten Roboter aus, gesteuert werden.</p> <p>Die Steuerungsabläufe können aufgezeichnet und zu einem späteren Zeitpunkt nachgestellt werden. Zwischen den zwei Robotern befindet sich das Bauteil, das zurzeit gescannt wird. Auf diesem werden einerseits die Scan-Linien der Roboter und andererseits die Fehlerwerte mittels einer Heatmap dargestellt. In der Android-App werden Heatmap und Roboter, einem Steuerungsablauf folgend, unter Verwendung von Augmented Reality auf ein Blatt Papier projiziert.</p>
-------------------	--


	<b>HTBLA Grieskirchen</b>	
	Fachrichtung:	<b>Informatik</b>

<b>Typische Grafik, Foto etc.</b> <b>(mit Erläuterung)</b>	<p>Der Screenshot zeigt die Roboter, welche gerade einem Steuerungsablauf folgen. Außerdem wird zwischen den Robotern die Heatmap veranschaulicht. In der rechten oberen Ecke sind UI-Elemente mit welchen man diverse Anzeigemöglichkeiten ein- beziehungsweise ausblenden kann. Am unteren Bildschirmrand werden Wiedergabeoptionen (Play/Pause) und Start/Stopp für die Aufnahmefunktion angezeigt.</p> 
---	--

<b>Teilnahme an Wettbewerben, Auszeichnungen</b>	<p>Falls der Wettbewerb „Autstanding“ aufgrund der aktuellen Corona-Krise stattfinden wird, werden wir an diesem teilnehmen.</p>
--	--

<b>Möglichkeiten der Einsichtnahme in die Arbeit</b>	<p>Der Quellcode dieser Diplomarbeit unterliegt auf Wunsch des Auftraggebers der Geheimhaltung und steht nur der Prüfungskommission zur Einsicht zur Verfügung. Eine Version der Diplomarbeit wird im Archiv verwahrt und ist für die öffentliche Einsichtnahme gesperrt.</p>
--	---

<b>Approbation</b> <b>(Datum/Unterschrift)</b>	Prüfer/Prüferin	Direktor/Direktorin
---	-----------------	---------------------


	HTBLA Grieskirchen	
	Fachrichtung:	Informatik

# DIPLOMA THESIS

## Documentation


<b>Author(s)</b>	<p>Jakob Deubler</p> <p>Aleks Dimitrov</p> <p>Christof Ecker</p>
<b>Form</b>	5BHIF
<b>Academic year</b>	2019/20
<b>Topic</b>	Digital Twin Ultrasonic-Inspection with AR
<b>Co-operation partners</b>	Fill GmbH

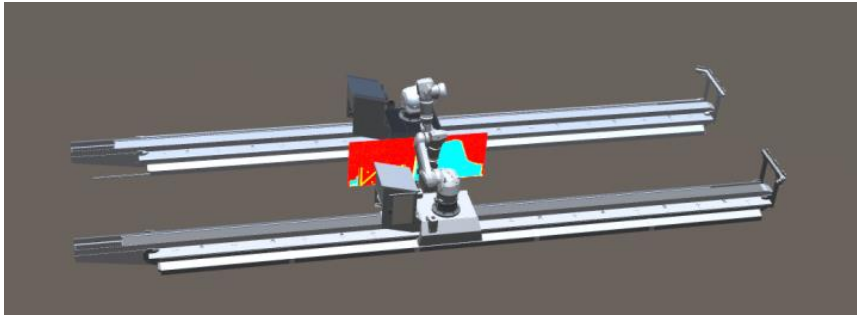
<b>Assignment of tasks</b>	<p>The goal of our thesis is to improve the illustration of the UltrasonicInspectionDualRobot, which checks for defects in the component. The results of the measurement should be displayed in form of a heatmap, located on the component.</p> <p>In the future the company Fill wants to utilize our project to visualize their robotic plant for potential customers on exhibitions and similar events.</p>
----------------------------	---

	<b>HTBLA Grieskirchen</b>	
	Fachrichtung:	<b>Informatik</b>

<b>Realisation</b>	<p>The models (robots, components, tools) were realized in the program Blender. The functionality was implemented in the development environment Unity. For the augmented reality projection, we used the vuforia kit. Also, we worked with the software unity collaborate for version controlling.</p> <p>In addition, we were operating with Google drive for any other necessary data exchanges. Furthermore, the communication between ourselves and our partner company took place on the platform Telegram.</p>
--------------------	---

<b>Results</b>	<p>The outcome of our thesis contains a windows program and an android application.</p> <p>With the windows program the functionality of the supersonic inception can be simply displayed. Additionally, the digital robots can be controlled by the control system of the actual robots.</p> <p>The control processes can be recorded and be replayed later on. Between both robots, the component that is being scanned at the moment, is located. On the component two different things are displayed. On the one hand there are the scan lines of the robots and on the other hand the error measurements in form of a heatmap. In the android app the heatmap and the robots, which trail the control processes, are projected on a sheet of paper using augmented reality.</p>
----------------	--

	<b>HTBLA Grieskirchen</b>	
	Fachrichtung:	<b>Informatik</b>

<b>Illustrative graph, photo (incl. explanation)</b>	<p>The screenshot shows one of the robots, which is currently following the control processes. Moreover, the heatmap is displayed between the robots. In the upper right corner, there are checkboxes to set the visibility of the various display options. On the bottom of the screen playback options (play/pause) and start/stop controls for the recording function are indicated.</p> 
--	---

<b>Participation in competitions Awards</b>	<p>If the contest “Autstanding” takes place despite the current corona crisis, we will participate in it.</p>
---	---

<b>Accessibility of diploma thesis</b>	<p>The source code of this diploma thesis is subject to secrecy due to a request of the client and is only available for inspection by the examination board. A version of this diploma thesis is kept in the archive and is not accessible to the public.</p>
--	--

<b>Approval (date/signature)</b>	<b>Examiner</b>	<b>Head of College/Department</b>
--------------------------------------	-----------------	-----------------------------------



# Inhaltsverzeichnis

<b>1</b>	<b>Einleitung .....</b>	<b>1</b>
1.1	Individuelle Aufgabenstellung .....	1
1.1.1	Jakob Deubler .....	1
1.1.2	Aleks Dimitrov .....	1
1.1.3	Christof Ecker .....	2
1.2	Auftraggeber .....	2
1.3	Ausgangslage .....	2
1.4	Anwendergruppe .....	3
1.5	Projektteam .....	3
<b>2</b>	<b>Technologien .....</b>	<b>4</b>
2.1	Unity .....	4
2.1.1	Unterstützte Zielplattformen .....	4
2.1.2	GameObjects .....	5
2.1.3	Prefabs .....	5
2.1.4	Unity Kamera .....	6
2.1.5	Sprites .....	6
2.2	C# .....	8
2.3	Unity Collaborate .....	8
2.4	Visual Studio .....	9
2.5	Blender .....	9
2.5.1	Object Mode .....	10
2.5.2	Edit Mode .....	10
2.5.3	Sculpt Mode .....	10
2.5.4	Texture Paint Mode .....	10
2.5.5	Rigging .....	10
2.5.6	Materialien .....	10
2.5.7	Licht .....	11
2.6	RabbitMQ .....	12
2.6.1	Producer .....	12
2.6.2	Exchange .....	13
2.6.3	Queue .....	13
2.6.4	Consumer .....	13
2.7	Vuforia .....	13
<b>3</b>	<b>Blender Modellierung .....</b>	<b>14</b>

3.1	Allgemein .....	14
3.2	Umsetzung.....	14
3.2.1	Modell.....	14
3.2.2	Materialien .....	15
3.2.3	Rig .....	18
3.2.4	Mobile Roboter Modell.....	19
<b>4</b>	<b>Kinematik .....</b>	<b>20</b>
4.1	Allgemein.....	20
4.2	TcpAxisPosition.....	20
4.3	Base .....	20
4.3.1	SetUp_RabbitMQ.....	20
4.3.2	Consumer_Received .....	21
4.4	RabbitMqReceiverMovement .....	21
4.4.1	AxisDataReceived.....	22
4.4.2	CalculatePosX.....	22
4.4.3	MoveFancy.....	22
4.5	RabbitMqReceiverRotations.....	23
4.5.1	Quaternion.....	23
4.5.2	AxisDataReceived.....	24
4.5.3	Rotate .....	26
4.6	Werkzeugwechsel.....	26
4.6.1	MeshRenderer .....	26
4.6.2	CheckTool .....	27
<b>5</b>	<b>Bauteildarstellung .....</b>	<b>28</b>
5.1	Heatmap .....	28
5.1.1	Positionen der Prüfwerte.....	28
5.1.2	Fehlerwerte .....	30
5.1.3	Farbe zu Fehlerwerten berechnen.....	30
5.1.4	Test mit GameObjects .....	33
5.1.5	Partikel System .....	33
5.1.6	Performance .....	34
5.2	Roboterprüfbahnen.....	34
5.2.1	Prüflinien.....	35
5.2.2	Einlesen der Daten.....	35
5.2.3	Test der Prüflinien.....	36
5.2.4	Line Renderer.....	36

5.3	Blender Modell .....	37
5.3.1	Initialisierung .....	37
5.4	File Browser .....	38
5.4.1	Verwendung des File Browsers aus dem Asset-Store.....	38
5.4.2	Automatischer Start.....	38
5.4.3	Auswahl der Dateien bei Auswahl des Ordners.....	39
<b>6</b>	<b>User Interface .....</b>	<b>41</b>
6.1	Rect-Transform .....	41
6.2	Canvas.....	41
6.2.1	Canvas Scaler .....	42
6.3	Checkboxen zum Ein- und Ausblenden der Ansichten.....	42
6.3.1	Checkbox für Roboterprüfbahnen .....	42
6.3.2	Checkboxen für Heatmap .....	43
6.4	Einstellen von Minimum und Maximum .....	45
6.4.1	Anzeigen der Schieberegler-Werte.....	45
6.4.2	Neuberechnung der Farben.....	45
6.5	Auswahl einer neuen Datei .....	46
<b>7</b>	<b>Integration der Test-Library .....</b>	<b>47</b>
7.1	JSON – JavaScript Object Notation .....	47
7.2	JSON Syntax .....	47
<b>8</b>	<b>Recording-Funktion .....</b>	<b>48</b>
8.1	Recording-Event (Base-Klasse) .....	48
8.2	Aufnahme-Button.....	49
8.3	Aufnahme-Skript (RecordingScript).....	50
8.3.1	AxisDataReceived.....	50
8.3.2	XML-File .....	50
<b>9</b>	<b>Datei einlesen (ReadData-Methode) .....</b>	<b>53</b>
9.1	Timer_Elapsed .....	54
<b>10</b>	<b>Pause Skript (PauseScript) .....</b>	<b>54</b>
10.1	DoPauseToggle().....	55
10.2	Roboterbewegung pausieren (PauseMovement) .....	56
10.3	Roboterbewegung starten (UnPauseMovement) .....	56
<b>11</b>	<b>Vuforia Basics .....</b>	<b>56</b>
11.1	Scoped repository.....	56

11.2	Vuforia-Account.....	57
11.3	License-Key hinzufügen .....	57
11.4	Vuforia Konfiguration .....	58
11.4.1	Verzögerte Initialisierung.....	58
11.4.2	Gerätekamera Modus.....	58
11.4.3	Vuforia-Datenbanken .....	58
<b>12</b>	<b>AR Projektion (Imagetarget und AR Kamera) .....</b>	<b>62</b>
12.1	AR Kamera .....	62
12.2	Imagetarget .....	62
12.2.1	Buttons mit AR anzeigen (ButtonPopup Skript).....	63
<b>13</b>	<b>Android-APK Erstellen.....</b>	<b>65</b>
13.1	JAVA_HOME setzen .....	65
13.1.1	Java Development Kit (JDK) .....	66
<b>14</b>	<b>Performance .....</b>	<b>67</b>
14.1	Unity Profiler .....	68
<b>15</b>	<b>Unity Probleme.....</b>	<b>70</b>
15.1	Unity rendert nicht .....	70
15.2	Unity-Versionen.....	70
15.2.1	Manifest.json .....	71
15.2.2	Package-Manager .....	71
15.2.3	Dependencies-Objekt .....	72
15.3	Problem bei AR-Kamera .....	73
15.4	Open Graphics Library (OpenGL).....	73
15.4.1	Unterschied zwischen OpenGL 2.1 und OpenGL 3.0 .....	73
<b>16</b>	<b>Danksagung .....</b>	<b>74</b>
<b>17</b>	<b>Statement des Auftraggebers .....</b>	<b>75</b>

# **1 Einleitung**

## **1.1 Individuelle Aufgabenstellung**

### **1.1.1 Jakob Deubler**

Die Aufgabe bestand darin, ein Bauteil, welches von den zwei Ultraschallprüfrobotern gescannt wird, darzustellen und drei verschiedene Ansichtsmöglichkeiten darauf zur Verfügung zu stellen. Die erste und auch wichtigste Ansicht ist die Heatmap. Es wird farbig angezeigt, an welchen Stellen die Prüfroboter Fehler im Bauteil festgestellt haben, und an welchen Stellen das Bauteil fehlerfrei ist. Die Daten hierzu werden aus mehreren Binärdateien, welche von der Firma Fill zur Verfügung gestellt wurden, eingelesen. Als zweite Ansichtsmöglichkeit sollen die Prüfbahnen, welche von den Robotern abgefahren werden, auf dem Bauteil dargestellt werden. Die nötigen Informationen werden aus einer XML-Datei ausgelesen, welche ebenfalls von der Firma Fill bereitgestellt wurde. Die letzte Ansicht zeigt das Blender-Modell des Bauteils. Dieses wurde bereits von Christof Ecker im Blender manuell bearbeitet und dann in Unity importiert und dargestellt.

### **1.1.2 Aleks Dimitrov**

Die Aufgabe von Aleks Dimitrov bestand aus zwei Aufgabenbereichen. Einerseits wurde eine Recording-Funktion, welche die Werte, die momentan von der echten Robotersteuerung kommen, in einer Datei gespeichert. Diese Datei, genannt Rezept, wird dann eingelesen, um den Robotern die benötigten Informationen weiterzugeben und schlussendlich dieses Rezept abzufahren. Außerdem soll eine Play-/Pause-Funktion programmiert werden. Die zweite Aufgabe bestand darin eine App, die mittels Augmented Reality die Roboter auf ein Blatt Papier projiziert, zu programmieren. Da das Robotermodell sehr detailliert war, traten Performance-Probleme auf. Ohne Performance-Verbesserungen sank die FPS Anzahl in der App auf 4-6 FPS. Die Projektion ruckelte und eine Performance-Verbesserung wurde vorgenommen, um den Kunden eine schlichte, stabile und schöne Projektion zu bieten. Schlussendlich stieg die FPS Anzahl auf 15-20 FPS.

### 1.1.3 Christof Ecker

Christof Ecker absolvierte ein Praktikum bei der Firma Fill. Während dieser Zeit begann er die Grundsteine des Projekts auszuarbeiten. Ein originalgetreues Modell des Roboters sollte in Blender realisiert werden. Hier waren drei Schritte von Relevanz. Das Modell des Roboters musste stückweise qualitativ angepasst werden, um später Performanceprobleme zu verhindern. Anschließend soll das Modell dem originalen Roboter entsprechend eingefärbt werden. Um eine Kinematisierung des Roboters zu ermöglichen muss ein Rig in das Mesh eingearbeitet werden. Dieses muss Gelenke in den Achsmittelpunkten bereitstellen, über welche anschließend die Achspositionen in Grad angepasst werden können. Zusätzlich soll auch die Kinematisierung des Roboters realisiert werden. Dazu muss ein RabbitMQ-Consumer (siehe 0) die Werte der echten Robotersteuerung, welche von der Firma Fill bereitgestellt wurden, einlesen. Das Modell soll seine aktuelle Position im Intervall von 50 ms an die aktuelle Vorgabe der Robotersteuerung anpassen, um so die Bewegungsabläufe der echten Anlage zu übernehmen. Zusätzlich müssen laufend Bauteile und Werkzeuge in Blender überarbeitet werden.

Christof Ecker übernahm zusätzlich die Aufgabe des Projektleiters.

## 1.2 Auftraggeber

Unsere Diplomarbeit wurde in Kooperation mit der Firma Fill aus Gurten, welche bereits über 850 Mitarbeiter zählt, absolviert. Das Unternehmen ist im Anlagenbau tätig und erzeugt individuelle Maschinen und Produktionsanlagen für ihre Kunden. Insbesondere arbeiteten wir mit der Abteilung Software 3 zusammen. Herr Gramberger und Herr Murauer waren unsere Ansprechpersonen bei der Firma, die uns einerseits die Aufgabenstellung der Diplomarbeit erklärten und uns andererseits während der technischen Umsetzung des Projekts unterstützten.

## 1.3 Ausgangslage

Die verwendete Codiersprache C# haben sich alle drei Kandidaten bereits über die letzten Schuljahre angeeignet. Jakob Deubler und Aleks Dimitrov haben auch schon Erfahrungen mit Unity gesammelt, was für Christof Ecker totales Neuland war. Da Unity allerdings sehr umfangreich ist, mussten alle Teammitgliedern ihr Wissen immer wieder auffrischen und erweitern. Ein großer Teil der Aufgaben, die während des Projektverlaufs anfielen, waren im Blender zu erledigen. Da keiner der Kandidaten sich bisher mit diesem Programm beschäftigt hatte, brachte sich Christof Ecker die benötigten Fähigkeiten per Online-Tutorials selber bei. Auch alle zur Umsetzung

des Augmented Reality Teils benötigten Kenntnisse eignete sich Aleks Dimitrov während des Projektverlaufs selbstständig an.

## 1.4 Anwendergruppe

Die Anwendergruppe der Diplomarbeit ist die Vertriebsabteilung der Firma Fill. Mit unserer Software ist eine einfache und moderne Präsentation des Ultraschallscan-Verfahrens vor Kunden und auf Messen möglich.

## 1.5 Projektteam

Name	E-Mail	Zuständigkeitsbereich
Jakob Deubler	<a href="mailto:jakob.deubler@gmail.com">jakob.deubler@gmail.com</a>	Bauteildarstellung
Aleks Dimitrov	<a href="mailto:aleks.dimitrov@hotmail.com">aleks.dimitrov@hotmail.com</a>	Recording-Funktion/AR-Projektion
Christof Ecker	<a href="mailto:ecker.christof01@gmail.com">ecker.christof01@gmail.com</a>	Projektleiter/Kinematik

## 2 Technologien

### 2.1 Unity

Unity ist eine Game Engine, welche erstmals 2005 veröffentlicht wurde. Hauptsächlich wird Unity zur Spieleentwicklung verwendet, wobei es auch zahlreiche andere Anwendungsmöglichkeiten gibt. Unity gibt Entwicklern, die Möglichkeit sowohl 2D, als auch 3D Spiele zu entwickeln. Als primäre Scripting API wird C# angeboten. Bevor C# die Hauptprogrammiersprache von Unity wurde, verwendete man die Sprache „Boo“. Diese wurde jedoch mit Unity 5 entfernt. Im Bereich der 2D-Spieleentwicklung unterstützt Unity das Importieren von Sprites (siehe 0) und stellt sehr leistungseffiziente „World renderer“ zur Verfügung. Die hohe Bedeutung von Unity lässt sich am hohen Marktanteil feststellen. Über die Hälfte aller Handyspiele wurden mit Unity erstellt. Außerdem wird es für die meisten Augmented und Virtual Reality Apps verwendet. Im Jahr 2018 wurden 60 Prozent aller AR/VR Anwendungen mit Unity entwickelt. Dies war auch einer der Gründe, warum wir uns für Unity entschieden haben. [1] [2]

#### 2.1.1 Unterstützte Zielplattformen

Ein großer Vorteil gegenüber anderen Entwicklungsumgebungen ist die hohe Anzahl an unterstützten Zielplattformen. Es werden unter anderem folgende Plattformen unterstützt:

- Desktop-Betriebssysteme
  - Windows, Mac und Linux
- Spielekonsolen
  - Nintendo Switch, Wii U, PlayStation 4, Xbox One und PS Vita
- Mobile-Betriebssysteme
  - iOS, Android und Windows Phone

Zusätzlich wird die Spiele-Engine auch abseits der Spieleentwicklung benutzt, zum Beispiel für Applikationen, welche von der leistungsfähigen 3D-Grafik profitieren. Zu nennen sind hierbei jegliche Arten von Simulationen und experimentellen Medienanwendungen. [1] [2]



### 2.1.2 GameObjects

Das Game-Objekt ist das wichtigste Konzept im Unity Editor. Jedes Objekt in der Szene ist ein „GameObject“. Dafür sind jedoch Eigenschaften notwendig, damit es zu einem Charakter, einer Umgebung oder einem speziellen Effekt umgewandelt wird.

Vordefinierte Komponenten geben dem Objekt Eigenschaften, welche zum Beispiel als Licht, Baum oder Kamera fungieren. Man kann sich das GameObject als einen leeren Topf und die Komponenten als Zutaten vorstellen. Je nachdem welche Art von Objekt erstellt werden soll, müssen verschiedene Kombinationen von Komponenten hinzugefügt werden. [3]

### 2.1.3 Prefabs

Unitys Prefab-System ermöglicht die Kreation, Konfiguration und das Speichern von kompletten Game-Objekten. Komponenten, Eigenschaften und angehängte GameObjects können ebenfalls eingestellt werden. Sie dienen als Template zur Erstellung neuer Instanzen. Sie werden erstellt, falls GameObjects wiederverwendet werden sollen. Im Ordner „Prefabs“ (siehe Abb 1: Prefabs) werden alle erstellten Prefabs abgespeichert. Änderungen werden automatisch an allen Instanzen des Prefabs übernommen, was zum zu einer großen Erleichterung bei Änderungen führt. Allerdings heißt dies nicht, dass die Instanzen gleicher Prefabs identisch sein müssen. Die Einstellungen können für individuelle Instanzen überschrieben werden. Sie werden auch für GameObjects, welche beim Starten der Szene noch nicht existieren und erst per Code zur Laufzeit erstellt werden, verwendet. Zum Beispiel: powerups, special effects oder Projektile. [4]



Abb. 1: Prefabs

### 2.1.4 Unity Kamera

Kameras sind dazu da, dem Endbenutzer ein Bild zu liefern. Es kann eine statische Kamera verwendet werden, um, wie in unserem Fall, einen bestimmten Teil der Szene aus einem festgelegten Blickwinkel darzustellen. Unity bietet aber auch Möglichkeiten, die Kamera zum Beispiel aus Sicht einer benutzergesteuerten Figur zu platzieren, "first-person" Perspektive genannt. Das Attribut Clear Flags legt fest, was an den Stellen passiert, in denen die Kamera ins Leere blickt. Die Standardeinstellung, genannt Skybox, legt eine blaue Himmellandschaft in den Hintergrund. Die zweite praxisrelevante Clear Flag ist die Option "Solid Color". Auch hier ist der Name für sich sprechend, leere Flächen werden mit einer ausgewählten Farbe gefüllt. [5]

### 2.1.5 Sprites

Sprites sind 2D-Grafik Objekte, welche durch Kombinieren und Modifizieren von Texturen einem 3D-Objekt ähneln können. Unity stellt vier wichtige SpriteTools zur Verfügung. [6]

#### 2.1.5.1 Sprite Creator

Der Sprite Creator dient als Platzhalter für Grafiken. Damit kann ohne dem eigentlichen Bild weitergearbeitet werden. [6]

#### 2.1.5.2 Sprite Editor

Mit dem Sprite Editor können mehrere Sprite-Grafiken aus einem größeren Sprite extrahiert werden. Diese bilden eine Reihe von Komponentenbildern (siehe Abb. 2: Sprite extrahiert). Dies wird zum Beispiel verwendet, wenn in einer Grafik mehrere Icons abgebildet sind, aber nur ein Icon benötigt wird. Im Editor kann ausgewählt werden auf wie viele Zellen oder Pixel die Grafik zugeschnitten werden soll. [6]

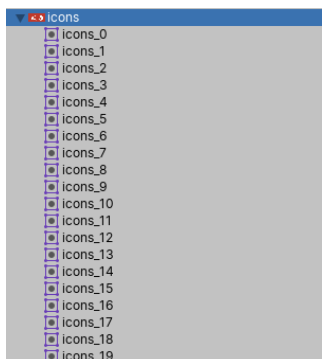


Abb. 2: Sprite extrahiert

### 2.1.5.3 Sprite Renderer

Sprites werden üblicherweise mittels des Sprite Renderers und nicht mit dem Mesh Renderer (siehe 4.6.1) gerendert. Er wird für die Darstellung von 2D- und 3D-Sprites in den Szenen verwendet. [6]

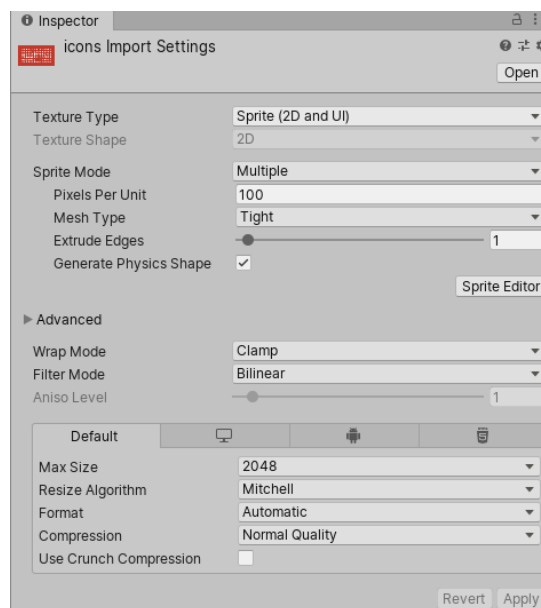
### 2.1.5.4 Sprite Packer

Der Sprite Packer wird verwendet, um die Leistung des Videospeichers für das Projekt zu optimieren. [6]

### 2.1.5.5 Sprite importieren

Sprites zählen im Unity-Projekt als Asset. Für das Importieren von Sprites stehen zwei Möglichkeiten zur Verfügung.

1. Die Grafik wird per Explorer in den Asset-Ordner kopiert. Unity erkennt diese Grafik automatisch und sie wird im Inspektor angezeigt.
2. Im Unity-Projekt unter Assets -> Import New Asset kann die gewünschte Grafik importiert werden. Hier muss das Bild manuell ausgewählt werden.



**Abb. 3: Sprite Inspektor**

Als „Texture Type“ (siehe Abb. 3: Sprite Inspektor) wird „Sprite (2D and UI)“ ausgewählt. Der Sprite Mode wird auf Multiple gesetzt, um aus einer Grafik mehrere Icons auszuschneiden. Mit dem Button „Sprite Editor“ öffnet sich der Sprite Editor (siehe 2.1.5.2) und die Grafik kann auf die gewünschten Icons zugeschnitten werden. [6]

## 2.2 C#

C# (gesprochen „C Sharp“) ist eine von Microsoft entwickelte Programmiersprache. Sie wurde hauptsächlich von Anders Hejlsberg designed und erstmals 2001 veröffentlicht. Es handelt sich um eine objektorientierte und typsichere Programmiersprache, welche ihre Wurzeln in der C-Sprachfamilie hat. Die Sprache zählt momentan zu den wichtigsten am Markt und hat auch weitere wichtige Sprachen, wie zum Beispiel Java und Swift, entscheidend beeinflusst. Historisch gesehen wurde C# ausschließlich für Windows entwickelt. Mittlerweile ist es jedoch auch möglich mit C# für macOS, iOS und Android zu entwickeln. [7]

## 2.3 Unity Collaborate

Unity Collaborate erleichtert es, gemeinsame Projekte zu speichern, zu teilen und zu synchronisieren. Collaborate ist Cloud-gehostet und direkt in Unity integriert [8]. Unter dem Tab „Collab History“ findet man alle Versionen auf einen Blick, welche jederzeit wiederhergestellt werden können.

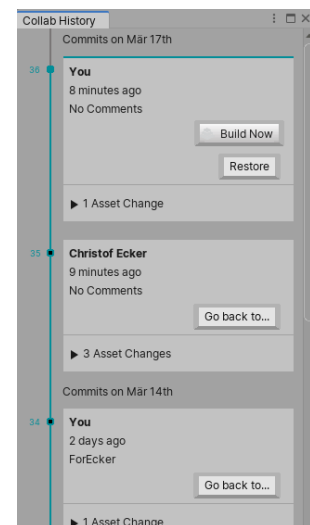


Abb. 4: Versionsgeschichte

Wurden Änderungen vorgenommen, die noch nicht geteilt wurden, sind sie in der rechten oberen Ecke unter dem Tab „Collab“ auffindbar. In dem Textfeld erklärt man, welche Änderungen vorgenommen wurden, um Teammitglieder auf dem neuesten Stand zu halten. Klickt man auf den Button „Publish now!“, so werden die Änderungen mit der Nachricht auf die Cloud hochgeladen. Im Tab wird ein großes grünes Häkchen angezeigt.

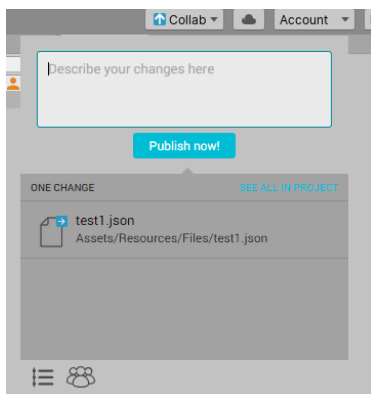


Abb. 5: Unity commit

## 2.4 Visual Studio

Microsoft Visual Studio ist eine von Microsoft angebotene Entwicklungsumgebung (IDE - „Integrated Development Environment“), welche für die Erstellung von Desktop- Anwendungen, Web Apps und Web Services verwendet wird. Visual Studio unterstützt 36 verschiedene Programmiersprachen, darunter auch C# (siehe 2.1.3), welche für die Diplomarbeit verwendet wurde. Visual Studio wird als eine sehr benutzerfreundliche Entwicklungsumgebung angesehen, da sie eine hohe Anzahl von unterstützten Funktionen aufweist. Wie jede andere IDE besitzt Visual Studio einen Code Editor, welcher Syntax-Highlighting, Code-Vervollständigung und IntelliSense, unterstützt. Des Weiteren ist natürlich in Visual Studio ein Debugger inkludiert, welcher nicht nur als source-level Debugger verwendet werden kann, sondern auch als machine-level Debugger nützlich ist. [9], [10]

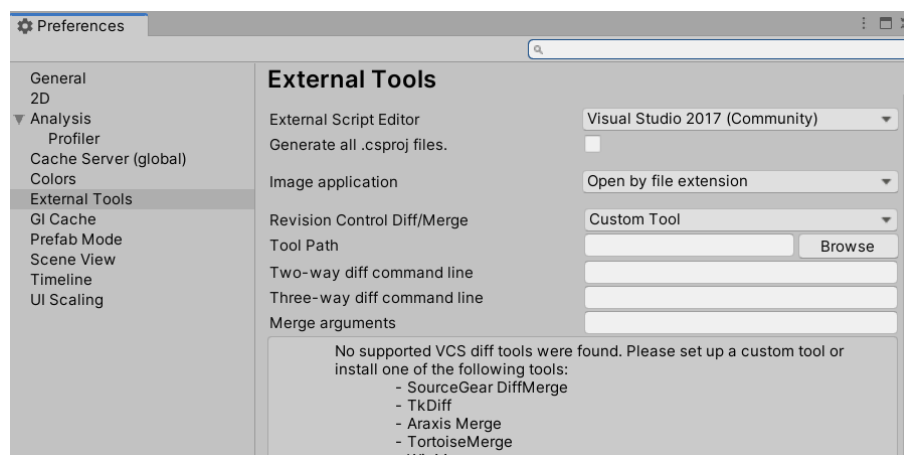


Abb. 6: Einbindung Visual Studio

## 2.5 Blender

Blender ist eine vielfältige kostenlose Software zur Gestaltung dreidimensionaler Szenen. Als Grundstruktur dienen Polygonnetze, die dreidimensionale Objekte, sogenannte Meshes, darstellen. Diese können mit Materialien versehen werden, um die Optik der Oberfläche anzupassen. Außerdem verwendet Blender seit der Version 2.8 Ray Tracing, einen Algorithmus der Lichtspiegelung, -brechung und -streuung berechnet, um höchst realistische Ergebnisse zu liefern.

Man arbeitet im Blender in verschiedenen Interaction Modes, welche die Grundfunktionen des großen Programms trennen. [11], [12]

### 2.5.1 Object Mode

Im Object Mode bildet man die Grundlage einer Blender-Modellierung. Hier werden die Mesh-Objekte erstellt und in der Szene verschoben, skaliert und rotiert.

### 2.5.2 Edit Mode

Im Edit Mode kann man in die Polygonnetze der einzelnen Objekte eingreifen und detailliert die Flächen, Kanten und Ecken der Meshes verändern.

### 2.5.3 Sculpt Mode

Im Sculpt Mode hat man verschiedene Pinsel zur Auswahl, mit denen man die Meshes wie Tonfiguren zurechtformen kann.

### 2.5.4 Texture Paint Mode

Im Texture Paint Modus weist man Meshes Texturen (siehe 2.5.6.5) zu, um ihre oberflächliche Erscheinung zu gestalten.

### 2.5.5 Rigging

Rigging wird benötigt, um Animationen oder Bewegungen eines Meshes nach Programmablauf zu ermöglichen. Als Grundlage benötigt man eine sogenannte "armature", eine Sammlung von Knochen. Diese sind an den Enden über Gelenke verbunden. Ein Rig kann man sich am besten am Beispiel eines menschlichen Skeletts vorstellen. Dabei sind die Knochen stabil und bewegen sich nur abhängig von Rotationen in den Gelenken. Will man ein Modell mit Rig per Code steuern, muss man später also auf die Gelenke zugreifen.

### 2.5.6 Materialien

Ein Material wird verwendet, um die optischen Eigenschaften des Materials, aus welchem ein Mesh gemacht ist, festzuhalten. Ganz einfach gesehen kann man Objekten damit eine "Farbe" zuweisen. Meistens sind für die Erscheinung eines Objekts vor allem die Oberflächenbeschaffenheiten, wie Farbe, Glanz und Reflektionen entscheidend. Blender Materialien bieten allerdings auch kompliziertere Effekte wie Transparenz, Lichtbrechungen und Volumenstreuung (siehe 0) an. [13]

#### 2.5.6.1 **Albedo Farbe**

Die Albedo Farbe eines Materials ist die grundsätzliche Farbe, die eine Oberfläche aufweist.

#### 2.5.6.2 **Metallische Beschaffenheit**

Die Metallische Beschaffenheit des Materials beeinflusst den Reflektionsgrad und das Verhalten der Oberfläche unter Lichteinstrahlung. Hat eine Oberfläche einen höheren metallischen Anteil, reflektiert sie mehr von der Umwelt, und es ist weniger von der Albedo Farbe zu erkennen.

#### 2.5.6.3 **Smoothness**

Die "Smoothness" eines Materials beschreibt wie glatt oder rau eine Oberfläche ist. Diese Eigenschaft ist nicht direkt sichtbar, sondern wird erst bei Lichteinstrahlung erkennbar. Je weicher eine Oberfläche ist, desto besser gebündelt werden Lichtstrahlen reflektiert und desto klarer sind Reflektionen.

#### 2.5.6.4 **Volumenstreuung**

Bei halbtransparenten Objekten dringen Lichtstrahlen ein, sie "hüpfen" im Objekt umher und treten an einer anderen Stelle wieder aus. Das Licht streut sich also im Objekt und es wirkt, als würde es selber leuchten. Diesen Effekt erkennt man besonders gut bei Menschen- oder Tierhaut, aber auch bei Trauben oder Tomaten.

#### 2.5.6.5 **Texturen**

Texturen sind Muster in Form von Bilddateien, die vielfach aneinandergereiht über die Oberfläche von Meshes gelegt werden. Sie sind so gestaltet, dass die Kanten der einzelnen Texturen aneinander anschließen und somit ein gesamtes Oberflächenbild erzeugen. Verwendung finden sie meistens in großen, eher hintergründigen Flächen. Ein hölzerner Fußboden mit einer bestimmten Musterung oder ein Kiesstrand sind gute Beispiele. [14]

#### 2.5.7 **Licht**

Licht ist ein sehr wichtiges Thema bei der 3D Modellierung. Jede Szene sollte mit Lichtquellen versehen werden, um sicherzustellen, dass die Meshes in der Szene in vollem Potential dargestellt werden. Unter- oder Überbelichtung können das Gesamtbild stark verschlechtern und müssen daher unbedingt vermieden werden. Dazu gibt es im Blender vier Möglichkeiten. [15]

#### 2.5.7.1 **Point Light**

Ein Point Light, ist wie der Name vermuten lässt, ein Punkt, der in alle Richtungen dieselbe Menge an Licht ausstrahlt. Verwendung findet das Point Light zum Beispiel bei Glühbirnen.

#### 2.5.7.2 **Spot Light**

Ein Spot Light stößt einen kegelförmigen Lichtstrahl aus der Spitze des Kegels in eine gewünschte Richtung aus. Es wird zum Beispiel bei Laternen verwendet. Die Spitze des Kegels muss sich dann in der Laterne befinden, und die Grundfläche des Kegels gibt Richtung und Streuungsgrad des Lichtstrahls an.

#### 2.5.7.3 **Area Light**

Ein Area Light simuliert Licht, das einer Oberfläche entstammt. Diese Oberfläche kann rechteckig, quadratisch, scheibenförmig oder elliptisch sein. Ein TV-Bildschirm oder eine LED-Leuchte sind typische Anwendungsbereiche.

#### 2.5.7.4 **Sun Light**

Ein Sonnenlicht bietet Licht in konstanter Intensität. Es entstammt einer Richtung, der Ursprung ist aber undefiniert und unendlich weit entfernt. Es eignet sich besonders gut für offene Plätze und Szenen im Freien.

### 2.6 **RabbitMQ**

RabbitMQ ist eine Open Source Middleware zur synchronen und asynchronen Übertragung von Nachrichten, basierend auf dem Advanced Message Queuing Protocol. Bei der Nachrichtenübermittlung gibt es vier Stationen: [16]

#### 2.6.1 **Producer**

Der Producer hat Daten gespeichert oder generiert laufend Daten, die weitergegeben werden sollen. Zusätzlich benötigt er eine Funktion, welche eine Nachricht zusammenstellt und sie dem Exchange übergibt.



### 2.6.2 Exchange

Der Exchange verteilt die Nachrichten vom Producer an die Queues. Abhängig davon wie die Nachrichten verteilt werden sollen, gibt es vier Arten von Exchanges. Im Direct Exchange gibt es nur eine Queue und in der Regel einen Consumer. Der Topic Exchange erweitert den Direct Exchange um die Möglichkeit, mehrere Queues anzusprechen. Es werden Keys verwendet, um den Queues die richtigen Pakete zuzuteilen. Der Fanout Exchange verteilt die Nachrichten einfach an alle verfügbaren Queues als Broadcast. Zuletzt gibt es noch den Header Exchange. Hier müssen die zu versendenden Nachrichten im Header ein Attribut enthalten, über welches der Exchange entscheiden kann an welche Queues die Information weitergeleitet werden soll.

### 2.6.3 Queue

Eine Queue erhält vom Exchange Nachrichten, die sie für den Consumer bereitstellen soll. Dies funktioniert wie ein Stapel, wobei die Reihenfolge, in der die Nachrichten eingehen, entscheidend ist. Fordert ein Consumer eine Nachricht von der Queue an, erhält er die unterste Nachricht. Diese wird dann aus der Queue entfernt.

### 2.6.4 Consumer

Der Consumer ist die empfangende Software, welcher sich bei einer Queue registriert und von dieser Nachrichten bezieht. Pro Queue darf es nur einen Consumer geben, da sonst nicht garantiert werden kann, dass die Daten richtig ankommen.

## 2.7 Vuforia

Für die Projektion mittels Augmented Reality wurde die Library „Vuforia“ verwendet. Die Vuforia-Library ist ein „augmented reality software development kit“ (SDK), welche die Erstellung einer Augmented-Reality-Applikation ermöglicht. Sie verwendet die Kamera des Geräts, um ebenflächige Bilder und 3D – Objekte in Echtzeit zu erkennen. Die Bildregistrierung ermöglicht den Entwicklern die Positionierung und Orientierung von virtuellen Objekten, 3D-Modellen und anderen Medien (zum Beispiel Videos und Bildern), in Relation zu den Echtzeit-Objekten. Das virtuelle Objekt detektiert die Position und die Orientierung des Bildes, auch Imagetarget genannt, in Echtzeit, sodass die Zuschauerperspektive auf das Objekt mit der Perspektive des Ziels übereinstimmt. Vuforia stellt eine API in C++, Java und C# an die Unity Game-Engine zur Verfügung. Dadurch wird die Verwendung auf den Zielplattformen, Android und iOS, unterstützt. [17]

## 3 Blender Modellierung

### 3.1 Allgemein

Die Grundvoraussetzung jedes Digitalen Zwillings ist ein Modell des echten Roboters, in diesem Fall des Ultrasonic Dual Inspection Robots. Aus mehreren Gründen wurde die Entscheidung getroffen, das Modell im Blender zu erstellen. Einer der Hauptgründe ist die Tatsache, dass man mit dem Blender erstellte Modelle gut in Unity importieren und verwenden kann. Dies gilt auch für das benötigte Rig, um das Modell später auch bewegen zu können. Des Weiteren handelt es sich bei Blender um eine kostenfreie Lösung.

### 3.2 Umsetzung

#### 3.2.1 Modell

Das Modell des Roboters besteht grundsätzlich aus zwei Sechs-Achs-Robotern, fixiert auf jeweils eigenen Verfahrsschlitten. Diese Verfahrsschlitten ermöglichen die horizontale Bewegung der Roboter auf den zwölfteinhalb Meter langen Schienen. Das gesamte Modell musste auf 2% der Originalgröße skaliert werden, um die Entwicklungsumgebungen nicht zu überlasten.



Abb. 7: Robotermodell

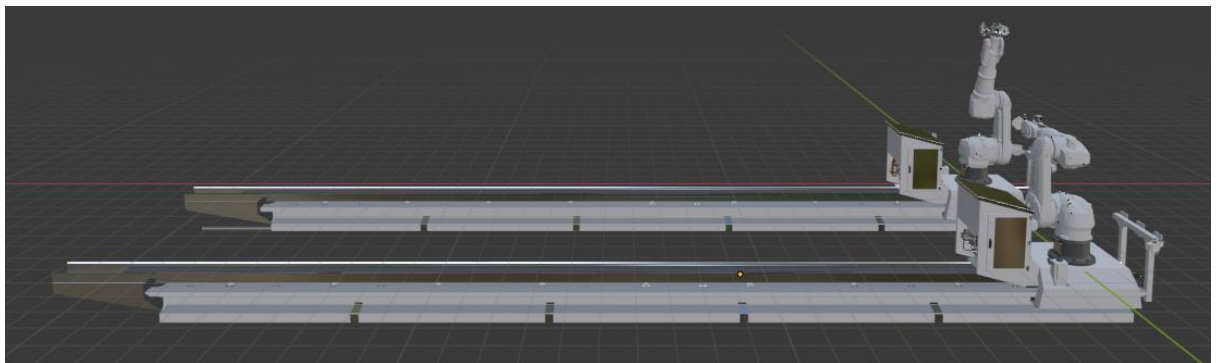
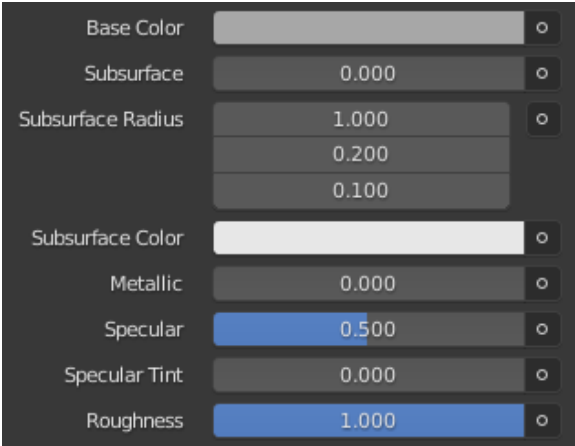
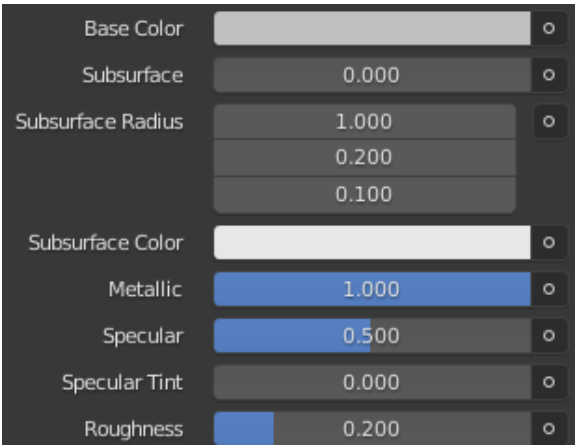
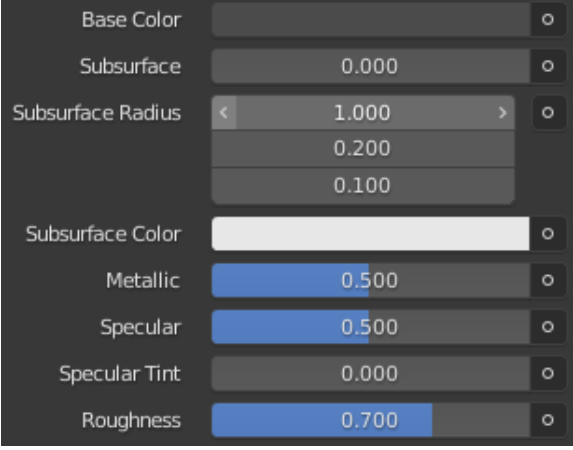
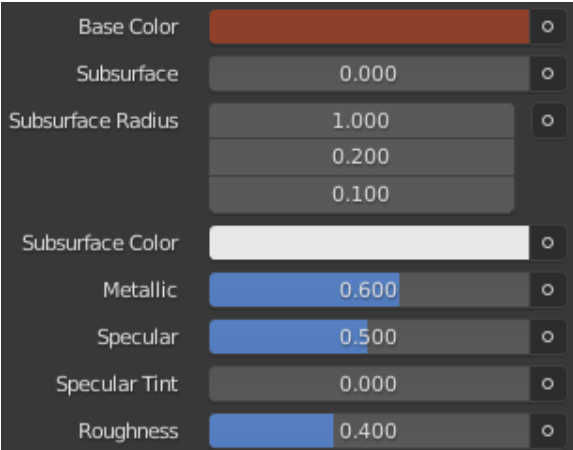
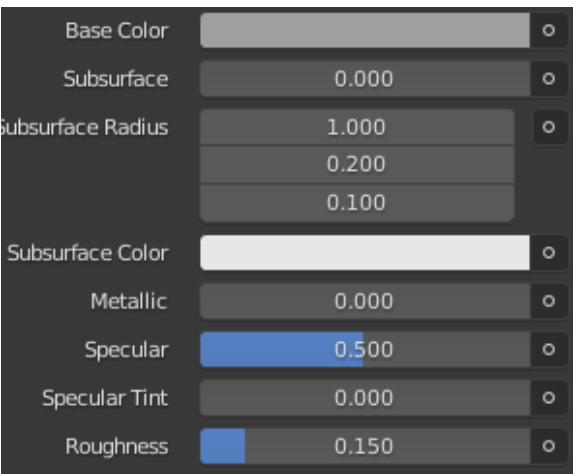


Abb. 8: Gesamtmodell

### 3.2.2 Materialien

Um die Oberfläche unseres Modells zu gestalten, wird an jeder Stelle des Roboters ein passendes Material hinterlegt. Dieses legt fest, welche Eigenschaften die Teile des Roboters haben sollen. So kann man zum Beispiel die grundsätzliche Farbe der Oberfläche festlegen, zusätzlich aber auch die Härte und die metallische Beschaffenheit. Die Optik des Roboters ergibt sich dann aus den Eigenschaften des Materials und den Lichtverhältnissen. Für unser Modell wurden folgende Materialien verwendet:

<p>Roboter Grau</p>	 <p><b>Abb. 9: Material Roboter Grau</b></p>
<p>Alu</p>	 <p><b>Abb. 10: Material Alu</b></p>

<p>Gummi</p>	 <p>The image shows the Blender Material properties panel for a material named 'Gummi'. The settings are as follows:</p> <ul style="list-style-type: none"> <li>Base Color: Default grey color swatch.</li> <li>Subsurface: 0.000</li> <li>Subsurface Radius: 1.000 (with a slider bar)</li> <li>Subsurface Color: Default white color swatch.</li> <li>Metallic: 0.500</li> <li>Specular: 0.500</li> <li>Specular Tint: 0.000</li> <li>Roughness: 0.700</li> </ul> <p>Abb. 11: Material Gummi</p>
<p>Kupfer</p>	 <p>The image shows the Blender Material properties panel for a material named 'Kupfer'. The settings are as follows:</p> <ul style="list-style-type: none"> <li>Base Color: Copper color swatch.</li> <li>Subsurface: 0.000</li> <li>Subsurface Radius: 1.000 (with a slider bar)</li> <li>Subsurface Color: Default white color swatch.</li> <li>Metallic: 0.600</li> <li>Specular: 0.500</li> <li>Specular Tint: 0.000</li> <li>Roughness: 0.400</li> </ul> <p>Abb. 12: Material Kupfer</p>
<p>Glas</p>	 <p>The image shows the Blender Material properties panel for a material named 'Glas'. The settings are as follows:</p> <ul style="list-style-type: none"> <li>Base Color: Default grey color swatch.</li> <li>Subsurface: 0.000</li> <li>Subsurface Radius: 1.000 (with a slider bar)</li> <li>Subsurface Color: Default white color swatch.</li> <li>Metallic: 0.000</li> <li>Specular: 0.500</li> <li>Specular Tint: 0.000</li> <li>Roughness: 0.150</li> </ul> <p>Abb. 13: Material Glas</p>

<p>Scharniere</p>	 <p>Abb. 14: Material Scharniere</p>
<p>Fill Logo</p>	 <p>Abb. 15: Material Fill Logo</p>

Die erstellten Materialien müssen an den entsprechenden Stellen in das Mesh eingearbeitet werden. Dazu werden im Edit-Mode (siehe 2.5.2) die einzelnen Flächen eines Meshes, die einem Material zugewiesen werden sollen, ausgewählt, und das gewünschte Material wird zugewiesen.

### 3.2.3 Rig

Das Rig eines Roboters besteht aus sechs Knochen, jeweils zugehörig zu einer Achse. Die Gelenke befinden sich im Mittelpunkt der zugehörigen Rotationsachse. Über sie können die Achsen des Digitalen Zwillinges später angesteuert werden, um die gewünschten Bewegungsabläufe auszuführen. Um ein Rig in unser Mesh einzubinden, sind zwei Schritte notwendig. Zuerst muss das Skelett unseres Roboters erstellt werden, und anschließend muss das Mesh des Roboters an die Knochen gebunden werden.

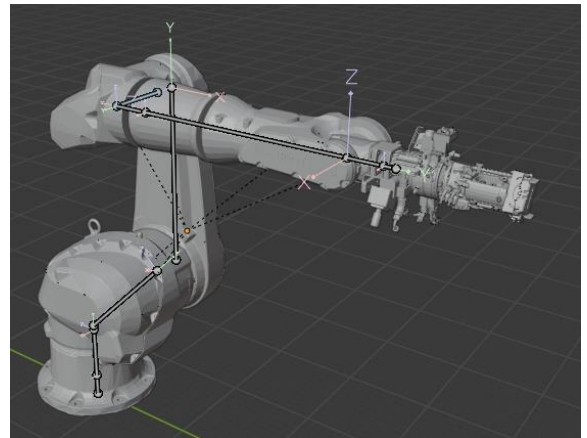


Abb. 16: Rig Detailansicht

#### 3.2.3.1 Skelett

Mit der Tastenkombination "Shift + A" erstellt man den ersten Knochen, welcher ins Zentrum der ersten Achse verschoben wird. Dabei ist es wichtig, die Gelenke an den Enden des Knochens genau mittig zur Achse zu platzieren, da diese später als Rotationspunkte dienen. Um weitere Knochen anzuknüpfen gibt es zwei Möglichkeiten. Als erste Option kann man am ersten Knochen das Gelenk wählen, welches an die zweite Achse anschließt. Man verwendet die Tastenkombination "Shift + E". Dadurch wird ein neuer Knochen an dieses Gelenk angehängt. Ist dieser richtig platziert, wiederholt man das, bis das Skelett vollständig ist. Da sich bei dieser Vorgehensweise aber immer zwei Knochen ein Gelenk teilen, trat bei der zweiten Achse ein Problem auf. Es musste auf die zweite Option zurückgegriffen werden. Um zwei Knochen, ohne sich ein Gelenk zu teilen, zu verbinden, wurde zuerst mit der Tastenkombination "Shift + A" ein neuer Knochen erstellt. Man wählt nun das Gelenk, an das der Knochen ursprünglich angeknüpft werden sollte. Unter dem Menüpunkt "Relations" findet man eine Box mit dem Namen "Parent". Klickt man darauf kann man das Gelenk, welches das Skelett fortsetzen soll, auswählen.

#### 3.2.3.2 Mesh an Knochen binden

Damit das Mesh die Bewegungen des Skeletts übernimmt, müssen die einzelnen Achsen an die zugehörigen Knochen geknüpft werden. Dazu wählt man zuerst im Object Mode das Mesh und anschließend im Pose-Mode den Knochen. Mit der Tastenkombination "Strg + P" öffnet man ein Popup mit den Parent-Optionen des Meshes. Hier kann man festlegen, dass es dem Knochen folgen soll.

### 3.2.4 **Mobile Roboter Modell**

Da die Performance der Handy-Applikation mangelhaft war, musste eine weniger detaillierte und somit performancetechnisch besser geeignete Version des Roboters erstellt werden. Als Ausgang dafür diente das ursprüngliche Blender Modell. Blender bietet einige Möglichkeiten, die sich hervorragend eignen, um das Modell des Roboters performance-technisch zu verbessern.

#### 3.2.4.1 **Decimate Geometry**

Die Funktion Decimate Geometry hat zwei Verwendungsmöglichkeiten. Die erste Möglichkeit vereint nahe beieinander liegende Punkte im Polygonnetz und reduziert so die Komplexität des Modells. Die zweite Möglichkeit erzielt ein ähnliches Ergebnis, indem es kleine, nahe beieinander liegende Flächen zusammenschließt. Decimate Geometry findet bei uns hauptsächlich am Modell des Roboters selber Gebrauch. Die vielen detaillierten Attribute werden vereinfacht oder entfernt.

#### 3.2.4.2 **Make Planar Faces**

Diese Funktion zielt auf große, flache Objekte in der Szene ab. Kleine Details auf diesen Objekten werden dann abgeflacht und eventuell sogar entfernt. Außerdem werden Flächen, die aneinander angrenzen, vereint. Diese Funktion findet vor allem an den Bahnen des Roboters Gebrauch, da diese grundsätzlich aus wenigen großen Flächen bestehen.

#### 3.2.4.3 **Delete Loose Geometry**

Diese Funktion entfernt alle losen Polygonnetz-Stücke, die beim Erstellen eines Modells entstehen und nicht gebraucht werden.

## 4 Kinematik

### 4.1 Allgemein

Da das Endprodukt ein Digitaler Zwilling sein soll, müssen die Bewegungsabläufe denen des echten Roboters entsprechen. Um dies zu ermöglichen, wurde die Robotersteuerung der Firma Fill erweitert. Das Programm sollte, zusätzlich zu den Ausgängen an die echten Roboter, eine Funktion bereitstellen, die dem Digitalen Zwilling über RabbitMQ die für Bewegungsabläufe nötigen Daten bereitstellt. Dazu werden im 50 ms Takt alle benötigten Werte im JSON-Format (siehe 7.2) bereitgestellt.

### 4.2 TcpAxisPosition

Die TcpAxisPosition-Klasse enthält alle Werte, die benötigt werden, um die momentane Position des Roboters zu bestimmen. Die RobotId bestimmt für welchen der zwei Roboter eine Instanz der Klasse vorgesehen ist. Die Winkelpositionen der sechs Roboterachsen werden in den doubles A1 bis A6 in Grad gespeichert. Der ComponentName gibt Auskunft darüber, welches Bauteil gerade bearbeitet wird. Welches Werkzeug dabei verwendet wird, ist im String ToolId hinterlegt. Dieser Wert wird später benötigt, um einen Werkzeugwechsel zu ermöglichen.

```
public class TcpAxisPosition
{
    public int RobotId { get; set; }

    public double A1 { get; set; }

    public double A2 { get; set; }

    public double A3 { get; set; }

    public double A4 { get; set; }

    public double A5 { get; set; }

    public double A6 { get; set; }

    public double XExt { get; set; }

    public double Feedrate { get; set; }

    public string ComponentName { get; set; }

    public string ToolId { get; set; }
}
```

Abb. 17: Klasse TcpAxisPosition

### 4.3 Base

#### 4.3.1 SetUp\_RabbitMQ

In der Methode SetUp\_RabbitMQ wird anfangs die Verbindung zur oben genannten Robotersteuerung aufgebaut. Dazu werden Adresse, Nutzerdaten und der Name der Queue benötigt.



```
private void SetUp_RabbitMQ()
{
    connection = new ConnectionFactory()
    {
        HostName = "10.1.33.19",
        UserName = "admin",
        Password = "admin"
    }.CreateConnection();
    model = connection.CreateModel();
    model.QueueDeclare(queueName, true, false, false, new Dictionary<string, object>() { { "x-message-ttl", 500 } });
}
```

Abb. 18: Methode SetUp\_RabbitMQ 1

Anschließend wird festgelegt, dass die Methode Consumer\_Received aufgerufen wird, wenn über die Queue neue Daten bereitgestellt werden.

```
var consumer = new EventingBasicConsumer(model);
consumer.Received += Consumer_Received;
model.BasicConsume(queueName, true, consumer);
}
```

Abb. 19: Methode SetUp\_RabbitMQ 2

#### 4.3.2 Consumer\_Received

Für beide Roboter gibt es jeweils einen EventHandler (RobotXAxisDataReceived). Wird dieser ausgelöst, bewegt sich der zugehörige Roboter an die übergebene Position. In der Methode Consumer\_Received werden zuerst die empfangenen JSON Daten in eine TcpAxisPosition-Klasse umgewandelt und die RoboterId wird überprüft. Anschließend wird abhängig von der RoboterId einer der oben genannten EventHandler mit der TcpAxisPosition-Klasse ausgelöst. Zum Verständnis der IF-Klausel siehe 9.1.

```
private void Consumer_Received(object sender, BasicDeliverEventArgs e)
{
    data = JsonConvert.DeserializeObject<TcpAxisPosition>(Encoding.UTF8.GetString(e.Body));

    if (!PauseScript.isPause)
    {
        if (data.RobotId == 2)
        {
            Robot2AxisDataReceived?.Invoke(this, new AxisPositionEventArgs(data));
        }
        else
        {
            Robot1AxisDataReceived?.Invoke(this, new AxisPositionEventArgs(data));
        }
    }
}
```

Abb. 20: Methode Consumer\_Received

### 4.4 RabbitMqReceiverMovement

Diese Klasse gibt es einmal für jeden Roboter. Sie ist für die Bewegung des gesamten Roboters auf der Schiene zuständig. Bei der Initialisierung wird die Methode AxisDataReceived an das RobotXAxisDataReceived-Event geknüpft. Das bedeutet, dass bei jedem Auslösen des Events die Methode AxisDataReceived aufgerufen wird.

```

void Start()
{
    if (!(this.name.EndsWith(" 1")))
    {
        Base.Instance.Robot1AxisDataReceived += AxisDataReceived;
    }
    else
    {
        Base.Instance.Robot2AxisDataReceived += AxisDataReceived;
    }
}

```

Abb. 21: RabbitMqReceiverMovement Start

#### 4.4.1 AxisDataReceived

Zuerst werden Input und Feedrate in Floats (Kommazahlen) umgewandelt und in globale Variablen übernommen. Anschließend wird die Methode CalculatePosX aufgerufen.

```

private void AxisDataReceived(object sender, AxisPositionEventArgs e)
{
    Input = Convert.ToSingle(e.Value.XExt);
    Feedrate = Convert.ToSingle(e.Value.Feedrate);
    CalculatePosX();
}

```

Abb. 22: Methode AxisDataReceived

#### 4.4.2 CalculatePosX

Die Variable PosX repräsentiert die nächste Stelle auf der Achse, an die der Roboter bewegt werden soll. Da sich das Koordinatensystem, das in der Robotersteuerung verwendet wird, von dem in Unity unterscheidet, kann PosX nicht einfach übernommen werden, sondern muss mit -1 multipliziert werden. Hinzu kommt, dass aufgrund der Skalierung des Roboters zusätzlich noch mit dem Faktor der Skalierung (0.02) multipliziert werden muss. Dies geschieht in der CalculatePosX Methode.

#### 4.4.3 MoveFancy

Die Methode MoveFancy wird in der Update Methode, also pro Frame einmal, aufgerufen. Da der Roboter nur an eine Position im Koordinatensystem, und nicht an eine Position an der X-Achse, bewegt werden kann, muss aus PosX zuerst ein Punkt im Koordinatensystem erstellt werden. Dazu wird die Variable NextPos, welche der Ausgangsposition des Roboters entspricht, um die gewünschte Position an der X-Achse verändert. Um eine flüssige Verschiebung des Roboters

zu ermöglichen, muss er die gewünschte Position genau dann erreichen, wenn sie wieder aktualisiert wird. Um diese Zeitspanne festzulegen wird `step` berechnet. Danach bewegt sich der Roboter über den berechneten Zeitraum gleichmäßig an die gewünschte Position.

```
private void MoveFancy()
{
    NextPos.x = PosX;
    step = 260f / (12f / Feedrate) * Time.deltaTime;
    transform.localPosition = Vector3.MoveTowards(transform.localPosition, NextPos, step);
}
```

Abb. 23: Methode MoveFancy

## 4.5 RabbitMqReceiverRotations

Diese Klasse gibt es einmal für jede Achse beider Roboter. Sie ist für die Rotationen an den jeweiligen Achsen zuständig. Die Funktionsweise ist ähnlich wie im Script `RabbitMqReceiverMovement`. Bei der Initialisierung wird ebenfalls die Methode `AxisDataReceived` an das `RobotXAxisDataReceived`-Event geknüpft. Beim Auslösen des Events wird sie aufgerufen und bekommt über die Parameter die gewünschte Achsrotation mitgegeben. Um ein Objekt in Unity zu rotieren, ist allerdings nicht nur ein Gradwert, sondern auch eine Quaternion nötig.

### 4.5.1 Quaternion

Quaternionen repräsentieren in Unity Rotationen. Sie sind sehr kompliziert und man verwendet sie daher sehr selten. Meistens nutzt man eine Quaternion, um eine neue Rotation auf Basis einer vorher bestehenden zu erstellen. Innerhalb einer Quaternion gibt es unter anderem die vier Parameter "x", "y", "z" und "w", welche die Rotation abspeichern. Man kann auf diese Werte ähnlich wie bei einem Array zugreifen. Hat man zum Beispiel eine Quaternion unter dem Variablennamen "quaternion" greift man auf "x" mit "quaternion[0]", auf "y" mit "quaternion[1]", zu. Zusätzlich wird im Parameter "eulerAngles" die Rotation in Form von Eulerschen Winkeln festgehalten. [18]

#### 4.5.1.1 Quaternion.Euler

Die Funktion `Quaternion.Euler` benötigt drei Parameter, nämlich die gewünschten eulerschen Rotationen um die Achsen X, Y und Z. Sie gibt eine Quaternion zurück, die alle benötigten Rotationswerte enthält.

## 4.5.1.2 Quaternion.RotateTowards

Die Funktion Quaternion.RotateTowards erstellt eine Rotation von einer Quaternion zu einer anderen und gibt diese wieder als Quaternion zurück. Zusätzlich zu den beiden Quaternionen muss man in den Parametern eine Variable angeben, die die Geschwindigkeit der Bewegung festlegt.

## 4.5.2 AxisDataReceived

Hier unterscheiden sich die RabbitMqReceiverRotations-Klassen, da abhängig von der Roboterachse Korrekturen vorgenommen werden müssen. Außerdem drehen sich die einzelnen Roboterachsen um unterschiedliche Achsen im Koordinatensystem.

In der Methode AxisDataReceived wird zuerst überprüft für welchen der beiden Roboter die eingehenden Werte vorgesehen sind. Anschließend wird mit der Funktion Quaternion.Euler die gewünschte Rotation gespeichert.

### 4.5.2.1 Achse 1

Die erste Roboterachse wird um die Y-Achse des Koordinatensystems gedreht. Daher werden die Rotationen der X- und Z-Achse immer auf die Ausgangsrotation gesetzt. Der eingehende Wert A1, welcher die gewünschte Rotation speichert, muss aufgrund der Differenzen zwischen dem Unity-Koordinatensystem und dem der Robotersteuerung um  $+90^\circ$  oder  $-90^\circ$  korrigiert werden. Dazu werden die Variablen degreeCorrectionRobX verwendet.

```
private void AxisDataReceived(object sender, AxisPositionEventArgs e)
{
    if (e.Value.RobotId == 2)
    {
        rotateTo = Quaternion.Euler(AxisXRotation, Convert.ToSingle(e.Value.A1) + degreeCorrectionRob1 * A1CorrectionRob1, AxisZRotation);
    }
    else
    {
        rotateTo = Quaternion.Euler(AxisXRotation, Convert.ToSingle(e.Value.A1) + degreeCorrectionRob2, AxisZRotation);
    }
}
```

Abb. 24: AxisDataReceived Achse 1

#### 4.5.2.2 Achse 2

Die zweite Roboterachse wird um die Z-Achse gedreht, daher werden die Rotationen der X- und Y-Achse immer auf die Ausgangsrotation gesetzt. Da die benötigte Korrektur des Eingangswertes für Achse 2 bei beiden Robotern 45° beträgt, braucht man keine IF-Klausel.

```
private void AxisDataReceived(object sender, AxisPositionEventArgs e)
{
    rotateTo = Quaternion.Euler(AxisXRotation, AxisYRotation, Convert.ToSingle(e.Value.A2) + A2Correction);
}
```

Abb. 25: AxisDataReceived Achse 2

#### 4.5.2.3 Achse 3/4/5

Die AxisDataReceived Methode der dritten, vierten und fünften Roboterachse sind identisch. Die Korrektur des Eingangswertes erfolgt, indem er von 360 subtrahiert wird. Die Rotation der Y- und Z-Achse bleibt unverändert.

```
private void AxisDataReceived(object sender, AxisPositionEventArgs e)
{
    var xAngle = 360 - Convert.ToSingle(e.Value.A3);
    rotateTo = Quaternion.Euler(xAngle, AxisYRotation, AxisZRotation);
}
```

Abb. 26: AxisDataReceived Achse 3

#### 4.5.2.4 Achse 6

Die sechste Roboterachse wird um die X-Achse gedreht. Die Korrektur des Eingangswertes erfolgt durch das Addieren von 90° zum Eingangswert. Die Methode checkTool wird im Punkt Werkzeugwechsel erklärt.

```
private void AxisDataReceived(object sender, AxisPositionEventArgs e)
{
    CheckTool(e.Value.ToolId);

    var yAngle = Convert.ToSingle(e.Value.A6) + 90;
    rotateTo = Quaternion.Euler(AxisXRotation, yAngle, AxisZRotation);
}
```

Abb. 27: AxisDataReceived Achse 6

### 4.5.3 Rotate

Hier wird die Achse mit der Funktion `Quaternion.RotateTowards` gedreht. In den Parametern werden die aktuelle Rotation, die gewünschte Rotation und die gewünschte Drehgeschwindigkeit übergeben. Aufgerufen wird `Rotate` in der Methode `Update`, also jedes Frame einmal.

```
private void Rotate()
{
    transform.localRotation = Quaternion.RotateTowards(transform.localRotation, rotateTo, speed);
}
```

Abb. 28: Methode `Rotate`

## 4.6 Werkzeugwechsel

Am Ende der sechsten Achse soll das gewünschte Werkzeug angezeigt werden. Dazu sind grundsätzlich beide verfügbaren Werkzeuge an die sechste Achse gebunden, allerdings wird abhängig von der `ToolId` nur das gewünschte Werkzeug sichtbar gemacht.

### 4.6.1 MeshRenderer

Der Mesh Renderer entnimmt dem Mesh Filter, welcher alle Meshes aus den Anhängen bereitstellt, ein Objekt. Dieses Objekt wird dann an der Position des zugehörigen GameObjects (siehe 0) gerendert. Im Mesh Renderer können außerdem einige Einstellungen über die Interaktion des Meshes mit Licht getroffen werden, zum Beispiel ob und wie das Objekt Schatten wirft. Um ein Werkzeug unsichtbar zu machen, muss man den zugehörigen MeshRenderer deaktivieren. Aktiviert man ihn wieder, wird das Werkzeug erneut sichtbar. In der Methode `getRenderers` werden die MeshRenderer in globalen Variablen gespeichert.

```
private void getRenderers()
{
    peRenderer = peSlider.GetComponentInChildren(typeof(MeshRenderer))[0] as MeshRenderer;
    sqRenderer = squirter.GetComponentInChildren(typeof(MeshRenderer))[0] as MeshRenderer;
}
```

Abb. 29: Methode `getRenderers`

#### 4.6.2 CheckTool

Wie bereits in 3.5.2.4 erwähnt wird die Funktion CheckTool in der AxisDataReceived-Methode der sechsten Achse aufgerufen. Abhängig von der ToolId werden die globalen Variablen currentStatePeRenderer und currentStateSqRenderer gesetzt. Abhängig von diesen Variablen werden in der Update Methode die MeshRenderer der Werkzeuge aktiviert oder deaktiviert.

```
private void CheckTool(string toolId)
{
    if (toolId == "PE_SLIDER" && !currentStatePeRenderer)
    {
        Debug.Log("AxisDataReceived Slider");

        currentStatePeRenderer = true;
        currentStateSqRenderer = false;

        currentTool = "PE_SLIDER";
    }
    else if (toolId == "SQUIRTER" && !currentStateSqRenderer)
    {
        Debug.Log("ToolReceived Squirter");

        currentStatePeRenderer = false;
        currentStateSqRenderer = true;

        currentTool = "SQUIRTER";
    }
}
```

Abb. 30: Methode CheckTool

```
void Update()
{
    try
    {
        Rotate();

        peRenderer.enabled = currentStatePeRenderer;
        sqRenderer.enabled = currentStateSqRenderer;
    }
    catch (Exception exc)
    {
        Debug.LogWarning(exc.Message);
    }
}
```

Abb. 31: RabbitMqReceiverRotations Achse 6 Update

## 5 Bauteildarstellung

Grundsätzlich geht es bei der Bauteildarstellung um die Projektion eines, von den zwei Prüfro-  
botern, gescannten Objektes. Je nachdem wie oft ein Bauteil gescannt wurde, werden verschie-  
dene Ansichten darauf zur Verfügung gestellt. Diese Ansichten werden dann zwischen den zwei  
Robotern dargestellt. Wird das zugehörige Fahrprogramm der Roboter ausgewählt, sieht es so  
aus, als würde die Anlage das Bauteil überprüfen (Simulation des Echtbetriebes der Roboter).

### 5.1 Heatmap

Die Heatmap zeigt ein Scanergebnis der zwei Roboter an. Die Daten dafür werden aus einem,  
von Fill zu Verfügung gestellten, Ordner ausgelesen. Der Ordner beinhaltet einerseits die Posi-  
tion und andererseits den Fehlerwert eines Prüfpunktes. Die hohe Anzahl der geprüften Positio-  
nen und der zusätzliche geringe Abstand zwischen den Positionen führt dazu, dass aus vielen  
kleinen Prüfpunkten eine große Prüffläche wird.

#### 5.1.1 Positionen der Prüfwerte

Die Positionen werden aus einer Binärdatei mit der Dateiendung „.pos“ (für Position) ausgele-  
sen. Da es nur sehr wenige Informationen über den Aufbau der Datei gab, war eine intensive  
Auseinandersetzung mit der Datei notwendig. Aufgrund der binären Speicherart der Datei lässt  
sie sich jedoch nicht in einem normalen Texteditor öffnen. Deswegen wurde ein  
Hex-Editor zum Betrachten der Datei verwendet. Diesen kann man in Notepad++ als Plugin  
installieren.

```
00 17 00 17 00 01 00 18 00 17 00 01 00 19 00 17
00 01 00 1a 00 17 00 01 00 1b 00 17 00 01 00 1c
00 17 00 01 00 1d 00 17 00 01 00 1e 00 17 00 01
```

Abb. 32: Ausschnitt der Binärdatei

In der Abbildung sieht man einen Ausschnitt des Position Files. Man erkennt dadurch schon ein  
gewisses Muster in der Datei. Wenn man alle angezeigten Nullen ausblendet, lässt sich eine  
Wiederholung von drei nah beieinanderliegenden Zahlen erkennen. Zum Beispiel sieht man in  
der ersten Zeile die Zahlen 17, 17, 1 gefolgt von 18, 17, 1. Diese Struktur ist in der ganzen Datei  
zu finden und lässt nur vage auf die Positionen der Fehlerwerte, welche immer aus den drei  
Achswerten (X, Y und Z) bestehen, schließen.



Ein detailliertes Auslesen der Positionen war nicht möglich. Deswegen stellte die Firma Fill Kontakt mit einem dänischen Partnerunternehmen her, welche den entscheidenden Hinweis zum Auslesen der Koordinaten lieferte. Bei der Binärdatei handelt es sich um eine Big-Endian Speicherung, das bedeutet, dass der signifikanteste Wert zuerst abgespeichert wird.

```
var allBytes = File.ReadAllBytes(pathToPositionFile);
for (int i = 0; i < allBytes.Length; i = i + 6)
{
    ushort xPosition = BitConverter.ToUInt16(new byte[2] { allBytes[i + 1], allBytes[i + 0] }, 0);
    double actualX = (xPosition + positionOffsetX) * positionFactorX;

    ushort yPosition = BitConverter.ToUInt16(new byte[2] { allBytes[i + 3], allBytes[i + 2] }, 0);
    double actualY = (yPosition + positionOffsetY) * positionFactorY;

    ushort zPosition = BitConverter.ToUInt16(new byte[2] { allBytes[i + 5], allBytes[i + 4] }, 0);
    double actualZ = (zPosition + positionOffsetZ) * positionFactorZ;

    Vector3 pos = new Vector3((float)actualX, (float)actualY, (float)actualZ);
}
```

Abb. 33: Einlesen der Positionen

Als erstes werden alle Bytes der Datei mit der Methode „File.ReadAllBytes()“ eingelesen. In der Schleife werden dann die einzelnen Achsenwerte aus dem Byte Array ausgelesen. Die ersten zwei Bytes enthalten die Werte für die X-Achse, die nächsten zwei die Werte für die Y-Achse und die letzten zwei die Werte für die Z-Achse. Aufgrund der Big-Endian Speicherung muss bei jedem Achsenwert das hintere Byte vor das vordere Byte gestellt werden. Mittels einem BitConverter wird das neu erstellte Byte Array dann in eine Zahl umgewandelt. Zu dieser Zahl muss jedoch noch die Verschiebung auf der jeweiligen Achse addiert werden, und sie muss mit dem Skalierungsfaktor multipliziert werden. Diese beiden Werte lassen sich aus der zum Bauteil zugehörigen XML-Datei auslesen.

### 5.1.2 Fehlerwerte

Die Fehlerwerte werden ebenfalls aus einer Binärdatei ausgelesen, diese besitzt jedoch die Dateiendung „.c3d“. Ähnlich wie zur Positionsdatei gab es auch zu dieser Datei wenig Information. Nach anfänglichen Schwierigkeiten war es aufgrund unserer Vorerfahrungen möglich, diese Datei auszulesen. Diese Datei verwendet nämlich ebenso die Big-Endian Speicherung.

```
byte[] allBytes = File.ReadAllBytes(path);
for (int i = 0; i < allBytes.Length - 1; i += 2)
{
    ushort readIn = BitConverter.ToUInt16(new byte[2] { allBytes[i + 1], allBytes[i] }, 0);
    int value = Convert.ToInt32(Math.Round(readIn * valueCoefficient));
}
```

Abb. 34: Einlesen der Fehlerwerte

Am Anfang wird die gesamte Datei mit der Methode „File.ReadAllBytes()“ eingelesen. Pro Fehlerwert sind zwei Bytes abgespeichert, und aufgrund der Big-Endian Speicherung muss das hintere Byte vor das vordere Byte gestellt werden. Mit dem BitConverter wird das neue Byte Array in eine Zahl konvertiert. Diese Zahl muss noch mit einem Koeffizienten multipliziert werden, welcher in einer anderen Datei im gleichen Ordner vorhanden ist. Um die Zahl als Integer verwenden zu können, wird sie mit der „Math.Round()“ Funktion gerundet und anschließend zu einem Integer konvertiert.

### 5.1.3 Farbe zu Fehlerwerten berechnen

Für die Darstellung der Heatmap wird zu jedem Prüfwert eine passende Farbe berechnet. Die Berechnung benötigt ein Minimum und ein Maximum in der Einheit Dezibel, welche vom Benutzer mittels eines Sliders über das User Interface eingestellt werden können. Dem Maximum wird die Farbe Rot, dem Minimum die Farbe Blau, zugewiesen. Prüfpunkte, die einen Prüfwert im hohen Dezibelbereich aufweisen und sehr nahe zum gesetzten Maximum liegen, deuten auf einen Fehler im Bauteil hin. Deswegen bekommen sie eine eher rötliche Farbe zugewiesen. Prüfpunkte mit einem eher niedrigen Prüfwert deuten auf die Korrektheit des Bauteils hin und bekommen deshalb eine eher bläuliche Farbe.

Zur technischen Umsetzung des Farbverlaufs von Blau auf Rot fanden sich auf der Webseite [19] wichtige Informationen. Auf der Seite konnte man gut erkennen, wie sich die einzelnen Werte bei der RGB (Red-Green-Blue) Kodierung veränderten, je nachdem wo sich die Farbe im Farbverlauf befindet. Weiters wird dort geschildert, welche einzelnen Farbwerte sich erhöhen oder verringern müssen, damit sich die Farbe wie gewünscht ändert. Um einen flüssigen Übergang von einer blauen Farbe (rot: 0, grün: 0, blau: 255) zu einer roten Farbe (rot: 255, grün: 0, blau: 0) zu erreichen, muss zuerst der Grünwert auf 255 erhöht werden.

Anschließend wird der Blauwert bis auf 0 verkleinert. Danach wird der Rotwert auf bis zu 255 erhöht und abschließend der Grünanteil wieder auf 0 reduziert. Dies ergibt dann die vier unten beschriebenen Bereiche und den gewünschten Farbverlauf.

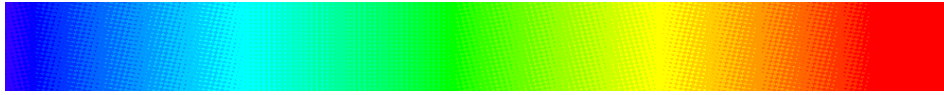


Abb. 35: Farbverlauf [40]

Die Berechnung beginnt, indem mithilfe der Spannweite zwischen dem Minimum und Maximum ein prozentualer Wert errechnet wird, welcher angibt, wo sich der Prüfwert zwischen diesen Zahlen befindet.

```
int range = max - min;  
double percentage = (double)(value - min) / range;  
Color color = new Color(0, 0, 255);
```

Abb. 36: Berechnung Farbe

Es gibt vier Bereiche in welche der prozentuelle Wert fallen kann:

### 5.1.3.1 Wert über 75 Prozent

In diesem Bereich werden der Rotwert auf 255 (100%) und der Blauwert auf 0 (0%) gesetzt. Anschließend wird der Prozentsatz um 75 verringert. Daraus ergibt sich, um wie viele Schritte der Grünwert unter 255 liegen muss. Die Größe dieser Schritte erhält man, indem der Maximalwert durch die Schrittzahl (25) dividiert wird, und er beträgt 10,2. Folglich wird der Grünwert in diesem Bereich kleiner je höher der Fehlerwert ist.

```
if (percentage > 0.75)  
{  
    double percentageOverColorGrade = Math.Round(percentage - 0.75, 2);  
    double greenValueToSubtract = percentageOverColorGrade * 10.2 * 100; //255 / 25 = 10.2  
    int greenValue = 255 - (int)greenValueToSubtract;  
    color = new Color(255, greenValue, 0);  
    return color;  
}
```

Abb. 37: Berechnung bei Farbwert über 75%

### 5.1.3.2 Wert zwischen 50 und 75 Prozent

In diesem Bereich werden der Grünwert auf 255 (100%) und der Blauwert auf 0 (0%) gesetzt. Der benötigte Rotwert wird ähnlich wie zuvor der Grünwert (siehe 5.1.3.1) berechnet. Dieses Mal wird jedoch der Rotwert höher, umso höher der Fehlerwert ist.

```
if (percentage > 0.5)
{
    double percentageOverColorGrade = Math.Round(percentag - 0.5, 2);
    double redValue = percentageOverColorGrade * 10.2 * 100;
    return color = new Color((byte)redValue, 255, 0);
}
```

Abb. 38: Berechnung bei Farbwert zwischen 50% und 70%

#### 5.1.3.3 Wert zwischen 25 und 50 Prozent

In diesem Bereich werden der Rotwert auf 0 (0%) und der Grünwert auf 255 (100%) gesetzt. Der Blauwert wird wie zuvor der Grünwert (siehe 5.1.3.1) berechnet.

```
if (percentage > 0.25)
{
    double percentageOverColorGrade = Math.Round(percentag - 0.25, 2);
    double blueValueToSubtract = percentageOverColorGrade * 10.2 * 100;
    int blueValue = 255 - (int)blueValueToSubtract;
    return color = new Color(0, 255, (byte)blueValue);
}
```

Abb. 39: Berechnung bei Farbwert zwischen 25% und 50%

#### 5.1.3.4 Wert unter 25 Prozent

In diesem Bereich wird der Rotwert auf 0 (0%) und der Blauwert auf 255 (100%) gesetzt. Der Grünwert wird wie der zuvor erklärte Rotwert (siehe 5.1.3.2) berechnet.

```
else
{
    double percentageOverColorGrade = Math.Round(percentag, 2);
    double greenValue = percentageOverColorGrade * 10.2 * 100;
    return color = new Color(0, (byte)greenValue, 255);
}
```

Abb. 40: Berechnung bei Farbwert unter 25%

#### 5.1.4 Test mit GameObjects

Anfangs wurde die Heatmap mit GameObjects (siehe 0) erstellt. Dazu wurde für jeden Fehlerpunkt ein „Sphere-GameObject“ generiert und an der richtigen Position initialisiert. Zusätzlich wurde das Objekt noch in der vorher berechneten Farbe eingefärbt. Bei diesem Test wurde jedoch klar, dass das Erstellen von Objekten an jeder Prüfstelle zu aufwändig wäre und zu einer extrem niedrigen Performance führt. Das Erzeugen der Heatmap dauerte einige Minuten, und auch nach der Erstellung war die FPS-Zahl sehr gering. Dabei war zu erkennen, dass das Einlesen der Prüfpositionen und der Prüfwerte korrekt war. Die Form des Bauteils konnte man aufgrund der Heatmap gut erkennen.

#### 5.1.5 Partikel System

Aufgrund der schlechten Performance bei der Erstellung der Heatmap mittels GameObjects, wurde das Unity-Particle-System als Alternative festgelegt. Das Partikel System unterstützt Entwickler, um bei der Darstellung trotz sehr vieler einzelner Punkte eine vernünftige Performance zu erhalten. Als Basis für das Partikel System dient ein einzelner Partikel. Dieser ist beim Erzeugen und während der Laufzeit sehr performanceschonend und lässt sich durch das Ändern seiner Eigenschaften individuell konfigurieren. Obwohl es normalerweise zur Erzeugung von visuellen Effekten, wie zum Beispiel Magie in einem Computerspiel, verwendet wird, lässt sich mit dem Partikel System auch die Heatmap relativ einfach erzeugen.

```
heatmap.Emit(pos, velocity, 15, timeToLive, color);
```

**Abb. 41: Partikel Erstellung**

Pro Prüfposition wird ein Partikel mithilfe der „Emit“ Funktion erstellt. Diese Methode benötigt als Parameter die Position, welche vorher aus der „\*.pos“ Datei ausgelesen wurde, und die Geschwindigkeit, welche auf einen leeren Vektor gesetzt werden kann. Der Partikel darf sich nicht bewegen, sondern muss auf seiner Position bleiben. Weitere Parameter sind die Größe des Partikels, die Zeit wie lange der Partikel bestehen bleibt, und die Farbe, welche bereits vorherberechnet wurde. Die Größe des Partikels wird auf 15 gesetzt, und als Überlebensdauer wird eine möglichst große Zahl verwendet, da der Partikel von Unity nicht zerstört werden soll, solange das Programm läuft.

### 5.1.5.1 Mehrere Partikel Systeme

Da es möglich sein soll mehrere Heatmaps pro Bauteil darzustellen, werden auch mehrere Partikel Systems benötigt. Theoretisch könnte man auch beide Heatmaps mittels eines gemeinsamen Partikel Systems erzeugen. Weil sich aber die einzelnen Partikel im Nachhinein nicht mehr identifizieren und nicht einer bestimmten Heatmap zuordnen lassen, würde es das spätere Ein- und Ausblenden unmöglich machen. Bei der Verwendung von mehreren Partikel Systemen tritt jedoch das Problem auf, dass ein GameObject nur ein Partikel System besitzen kann. Dieses Problem lässt sich umgehen, indem pro Heatmap ein neues GameObject als Kind des eigentlichen Objekts erstellt wird, und an jedes einzelne ein Partikel System angehängt wird.

```
GameObject heatmapGameObject = new GameObject("Heatmap: " + relatedMeasurements);  
heatmapGameObject.transform.parent = this.gameObject.transform;  
ParticleSystem heatmap = heatmapGameObject.AddComponent<ParticleSystem>();
```

Abb. 42: Erstellung Partikelsystem

### 5.1.6 Performance

Trotz der Performanceoptimierung mit Hilfe des Partikel Systems nahm das Erzeugen der Heatmap noch immer viel Zeit in Anspruch. Des Weiteren war die FPS-Anzahl nicht ausreichend um ein flüssiges Programm zu gewährleisten. Um die Performance weiter zu steigern, wurde beim Erstellen eine „counter“ Variable eingefügt. Diese zählt, wie viele Partikel erstellt werden sollen und bestimmt, dass nur jeder dritte Partikel erzeugt wird. Aufgrund der gigantischen Menge an Prüfpositionen lässt sich mit dieser Optimierung extrem viel Laufzeit einsparen. Am endgültigen Produkt merkt man diese Änderung nicht, weil eng angrenzende Prüfwerte enorm ähnlich sind und sehr nah beieinander liegen.

## 5.2 Roboterprüfbahnen

Eine weitere Ansichtsmöglichkeit auf das Bauteil sind die Roboterprüfbahnen, welche die Anlage abfahren, um Fehler im Bauteil festzustellen. Die Daten dafür werden aus einer von Fill zu Verfügung gestellten Datei ausgelesen.

### 5.2.1 Prüflinien

Die Daten werden aus einer XML-Datei ausgelesen. In dieser XML-Datei befinden sich Prüfpunkte, welche zu einer Linie verbunden werden. Für jede Linie gibt es in der XML-Datei einen Knoten, genannt „Line“, welcher ein Index-Attribut besitzt. Die Linien besitzen als Unterelemente jeweils mehrere Punkte, welche nochmals mehrere Knoten untergeordnet haben. Jeder Punkt besteht unter anderem aus einem X, Y und Z Wert, welche die Position des Punktes beschreiben (siehe Abbildung).

```
<Line index="1">
  <Point index="1">
    <X>2970.9845</X>
    <Y>1.0502</Y>
    <Z>-0.5078</Z>
    <A>-13.1277</A>
    <B>10.5326</B>
    <C>5.2336</C>
    <X_EXT1>4454.624016</X_EXT1>
    <X_EXT2>4254.624016</X_EXT2>
  </Point>
```

Abb. 43: Ausschnitt XML-Datei

### 5.2.2 Einlesen der Daten

Zum Einlesen der Daten wird die Klasse „XMLDocument“ verwendet. Ein neues „XMLDocument“ Objekt wird erzeugt, und mittels der Methode „Load“ wird die Datei geladen. Diese Methode besitzt als Parameter den Dateipfad zum XML-Dokument. Anschließend wird die erste Linie in der Datei als Knoten ausgewählt und die „ChildObjects“ von diesem ermittelt. Diese Unterelemente sind die einzelnen Punkte einer Linie. Die Punkte bestehen wiederum aus den einzelnen Achsenwerten, welche ebenfalls mit der Methode „ChildNodes“ erhalten werden. Um jeden Achsenwert aus dem Achsen-Knoten zu erhalten, muss der innere Text jedes Knotens verwendet werden. In der XML-Datei werden Kommazahlen mittels eines Punktes abgespeichert. Aufgrund der Tatsache, dass C# Kommazahlen mittels eines Kommas verwendet, muss jeder Punkt mit einem Komma ersetzt werden. Um die endgültige Position jedes Punktes zu erhalten, wird jeder Achsenwert noch zusätzlich mit der Verschiebung auf der jeweiligen Achse zusammengerechnet und mit dem Skalierungsfaktor multipliziert, um die richtigen Proportionen zum Roboter zu erhalten. Alle Punkte einer Linie werden als Vector3 Objekte in einer Liste, welche später zur Darstellung der Linien benötigt wird, gespeichert.

```
foreach (XmlNode pointNode in lineNode.ChildNodes)
{
    XmlNode pointNodeX = pointNode.ChildNodes[0];
    XmlNode pointNodeY = pointNode.ChildNodes[1];
    XmlNode pointNodeZ = pointNode.ChildNodes[2];

    Vector3 pointPos = new Vector3(
        Convert.ToSingle((float.Parse(pointNodeX.InnerText.Replace('.', ',')) + offsetX) * scaleValue),
        Convert.ToSingle((float.Parse(pointNodeY.InnerText.Replace('.', ',')) + offsetY) * scaleValue),
        Convert.ToSingle((float.Parse(pointNodeZ.InnerText.Replace('.', ',')) + offsetZ) * scaleValue));
}
```

Abb. 44: Einlesen der Prüfbahnen

### 5.2.3 Test der Prüflinien

Um das Einlesen der Prüflinien auf Korrektheit zu prüfen, wurden anfangs die Roboterprüfbahnen mittels der von Unity zur Verfügung gestellten Funktion „Debug.DrawLine“ veranschaulicht. Hierbei werden zwei Punkte mitgegeben, welche mit einer Linie verbunden werden. Dies wurde umgesetzt, indem immer zwei Punkte einer Linie gespeichert wurden: einerseits der aktuelle Punkt und andererseits der vorherige Punkt. Mittels der erzeugten Linien konnte man die Form des Bauteils erkennen, was auf das richtige Einlesen der Daten deutete. Das Problem der „Debug.DrawLine“ Methode ist jedoch, dass die Linien, welche mittels dieser Funktion erzeugt werden, nur in der „Scene-View“ von Unity dargestellt werden und nicht in der „Game-View“. Dies führt dazu, dass die Linien nur während der Entwicklung des Programmes und nicht bei der endgültigen Benutzung sichtbar sind. Aufgrund dieses Problems wurde die Erstellung der Roboterprüflinien auf die Erzeugung mittels eines „LineRenderer“ umgestellt.

### 5.2.4 Line Renderer

Der Line Renderer ist eine von Unity entwickelte Komponente, welche zwischen zwei oder mehreren Punkten eine gerade Linie zeichnet. Der Line Renderer kann verwendet werden um einfache Linien aber auch komplexe Objekte zu zeichnen.

#### 5.2.4.1 Voreinstellung

Bei der Verwendung wurde zuerst das GameObject um eine Line Renderer Komponente ergänzt. An diesem GameObject ist auch das C#-Skript angehängt, welches für die Darstellung der Linien zuständig ist. Am Beginn des Skripts wird der Line Renderer zuerst initialisiert und konfiguriert. Hier wird die Dicke der einzelnen Linien und das Material, aus welchem die Linien bestehen, festgelegt. Als Material wird das von Unity vordefinierte Default-Line Material verwendet, welches einfach in Form einer globalen Variable von Unity ans Skript übergeben wird. Diese Einstellungen sind für die Benutzung des Line Renderers nötig.



#### 5.2.4.2 Erzeugen der Prüflinien

Beim Erzeugen der Linien wird die beim Einlesen der Daten erstellte Liste von Punkten wieder aufgegriffen. Durch diese Liste wird iteriert und für jeden Punkt wird dem Line Renderer mittels der „SetPosition“ Methode dieser Punkt gesetzt. Zusätzlich wurde die Variable „positionCount“ des Line Renderers um eins pro Punkt erhöht. Dies ist für das richtige Erstellen der Linien essenziell, weil sonst der Line Renderer mehr Punkte zugeordnet bekommt als in der „positionCount“ Variable festgelegt wurde.

```
foreach (Vector3 position in pointPositions)
{
    renderer.positionCount += 1;
    renderer.SetPosition(counter, position);
    counter++;
}
```

Abb. 45: Erzeugen der Prüfbahnen

### 5.3 Blender Modell

Als dritte Ansichtsmöglichkeit des gescannten Bauteils soll das zugehörige Blender Modell erzeugt werden. Dieses Modell wurde zuvor in Blender bearbeitet und im Unity Projektordner abgespeichert. Beim Importieren der Blender Datei in Unity ist jedoch zu beachten, dass die Koordinatensysteme von Unity und Blender nicht ident sind und deswegen spezielle Anpassungen im Blender nötig sind. Mittels einer globalen Variable wird die Blender-Datei an das Skript übergeben.

#### 5.3.1 Initialisierung

Die Initialisierung erfolgt mit der von Unity bereit gestellten „Instantiate“ Methode. Die Funktion benötigt als Parameter das Blender Objekt, die Position und die Rotation. Davor wird die Rotation, mithilfe der vorher eingelesenen Daten, berechnet. Die Position an welcher das Bauteil generiert wird, wurde ebenfalls schon zuvor ermittelt.

```
private void LoadBlenderFile(string pathToBlenderFile)
{
    Quaternion rotation = Quaternion.Euler(new Vector3(rotationOffset.x, rotationOffset.y + 180, rotationOffset.z + 180));
    component = Instantiate(component, spawnPosition, rotation);
}
```

Abb. 46: Initialisierung des Blender Modells

Weil jedoch bei den, durch die Heatmap und Roboterprüfbahnen erhaltenen, Prüfflächen bereits die Form des Bauteils sehr gut erkennbarer, wurde diese Ansichtsmöglichkeit aus dem endgültigen Produkt entfernt.

## 5.4 File Browser

Für die Auswahl des Ordners, welcher die XML-Datei für die Roboterprüfbahnen und die Daten für die Heatmap beinhaltet, wird ein File Browser benötigt.

### 5.4.1 Verwendung des File Browsers aus dem Asset-Store

Weil die Programmierung eines eigenen File Browsers sehr viel Aufwand bedeutet hätte und den Rahmen der Diplomarbeit gesprengt hätte, wurde für das Projekt ein File Browser aus dem Unity Asset-Store verwendet. Es wurde der „Runtime File Browser“ von Süleyman Yasir Kula ausgewählt, weil dieser genau die Anforderungen erfüllt und zusätzlich noch kostenlos ist.

## Runtime File Browser



Süleyman Yasir Kula



5 | 44 Reviews

**FREE**

Abb. 47: Unity Asset: File Browser

Um eine Erweiterung aus dem Unity Asset-Store in das eigene Unity Projekt zu implementieren, muss zuerst online auf der Webseite des Asset-Store [20] das gewünschte Asset zu den eigenen Assets hinzugefügt werden. Wichtig ist hier, mit dem eigenen Unity-Account angemeldet zu sein. Danach kann in Unity unter „Window/Package Manager“ das Asset gedownloadet und importiert werden. Zeitgleich muss man mit demselben Account wie beim Hinzufügen auf der Webseite angemeldet zu sein.

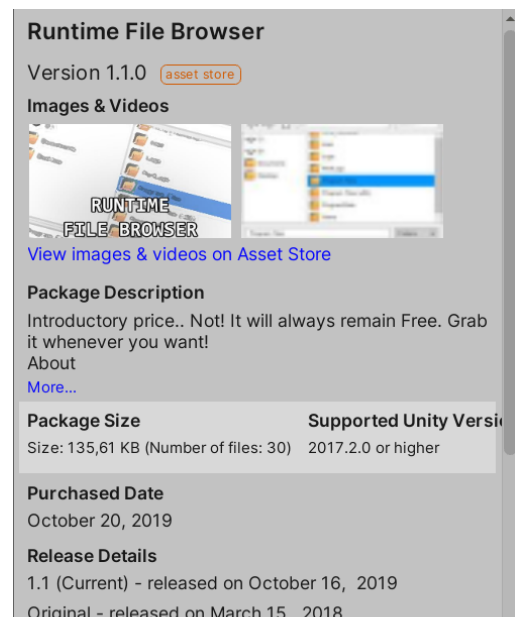


Abb. 48: Importieren des File Browsers

### 5.4.2 Automatischer Start

Da jedes Mal, wenn die Applikation gestartet wird, ein Bauteil angezeigt werden soll, muss auch ein automatischer Start des File Browsers gewährleistet werden. Um den File Browser immer zu aktivieren, wenn das Programm gestartet wird, wird in der „onStart“ Methode des C#-Skripts „StartCoroutine(ShowLoadDialog())“ aufgerufen. Bei „StartCoroutine“ handelt es sich um eine von Unity bereit gestellte Methode. Vereinfacht kann man sagen, dass aufgrund dieser Funktion

Unity nicht strikt auf ein Rückgabeobjekt wartet, sondern weiterläuft. Somit kann man später auf Änderungen des File Browsers, wie zum Beispiel die Auswahl eines Ordners, reagieren. „StartCoroutine“ benötigt als Parameter die Methode, die aufgerufen werden soll. Wichtig ist, dass diese Methode ein „IEnumerator“ Objekt retourniert und dass „return-statement“ das Schlüsselwort „yield“ verwendet. Der File Browser wird mit der Funktion „FileBrowser.WaitForDialog“ aufgerufen. Mittels der beim Aufruf benötigten Parameter erfolgt die Einstellung, dass nur Ordner ausgewählt werden dürfen, und es keinen standardmäßig festgelegten Dateipfad gibt. Zusätzlich wird noch der Titel des File Browsers und der Text des Buttons, welcher für die endgültige Auswahl des Ordners verwendet wird, festgelegt. Anschließend wird die „FileBrowser.Success“ Property überprüft, welche true wird, sobald ein Ordner ausgewählt wird.

```
IEnumerator ShowLoadDialogCoroutineXML()
{
    yield return FileBrowser.WaitForLoadDialog(true, null, "Load Scanned Object Folder (must include Nk3-Folder and XML-File)", "Load");
    if (FileBrowser.Success) { ... }
}
```

**Abb. 49: Automatischer Start des File Browsers**

### 5.4.3 Auswahl der Dateien bei Auswahl des Ordners

Sobald ein Ordner ausgewählt wurde, kann mittels der „FileBrowser.Result“ Property der Dateipfad zum ausgewählten Ordner ausfindig gemacht werden. Da jedoch der Pfad zu den einzelnen Dateien benötigt wird, werden zuerst die Dateipfade aller Dateien ermittelt.

#### 5.4.3.1 Auswahl der XML-Datei

Um den Pfad zur XML-Datei zu erhalten, werden alle Pfade auf ihre Dateiendung überprüft. Falls diese „.xml“ lautet, wird mit diesem Dateipfad die Methode zum Erstellen der Roboterprüflinien aufgerufen.

#### 5.4.3.2 Auswahl des NK3-Ordners

Die Daten für die Heatmap befinden sich in zwei unterschiedlichen Dateien. In einer Datei liegen die Koordinaten der Prüfungen und in einer anderen die zugehörigen Prüfwerte. Beide Dateien liegen jedoch in einem gemeinsamen Ordner, dem NK3-Ordner, welcher immer die Dateierweiterung „.nk3“ aufweist. Nach der Auswahl eines Ordners durch den File Browser wird in diesem nach einem Ordner mit der Dateierweiterung „.nk3“ gesucht. Dies erfolgt, indem zuerst alle Unterordner des ausgewählten Ordners mittels der „Directory.GetDirectories()“ ermittelt werden. Anschließend wird überprüft, ob ein Folder mit der benötigten Dateierweiterung endet, was mit der Methode „.EndsWith(„.nk3“)“ ermittelt wird. Wird ein Ordner gefunden, wird in diesem überprüft, wie viele verschiedene Bauteilprüfungen und dadurch verschiedene Heatmaps vorhanden sind. Innerhalb des NK3-Ordners befindet sich pro Bauteilprüfung ein „Acquisition“ Ordner, welcher einen Index im Dateinamen enthält. Die Anzahl an zu erstellenden Heatmaps wird ermittelt, indem der Pfad zur Positionsdatei überprüft wird. Pro existierendem Pfad muss eine Heatmap angelegt werden.

```
string[] item = Directory.GetDirectories(directory);
foreach (string nk3Directory in item) // nk3Directory == nk3 Ordner
{
    if (nk3Directory.EndsWith(".nk3"))
    {
        int counter = 0;
        while (true)
        {
            string path = nk3Directory + @"\Patch 1\Acquisition " + counter + @" Data\kuka.pos";
            if (File.Exists(path)) ...
            else
            {
                break;
            }
            counter++;
        }
    }
}
```

Abb. 50: Automatische Auswahl des NK3-Ordners

Mit den ermittelten Pfaden zur Positionsdatei und der zugehörigen Fehlerwertdatei werden die Funktionen zur Erstellung der Heatmap aufgerufen. Zusätzlich muss noch die Methode zum Festlegen der Voreinstellungen, wie zum Beispiel der Verschiebung oder Skalierung auf den einzelnen Achsen, verwendet werden.

```
pathToPosFile = nk3Directory + @"\Patch 1\Acquisition " + counter + @" Data\kuka.pos";
pathToC3DFile = nk3Directory + @"\Patch 1\Acquisition " + counter + @" Data\Cartography Data\A0.c3d";

InitializeData(nk3Directory + @"\Patch 1\Mechanical Data\test.mhf");
LoadMeasurements(pathToC3DFile);
LoadNK3File(pathToPosFile);
```

## 6 User Interface

Für das User Interface wurde die UI Library von Unity verwendet. Sie ermöglicht eine relativ einfache Erstellung einer Benutzeroberfläche mittels vordefinierten Unity UI-Komponenten, wie zum Beispiel Buttons oder Textfeldern. In der Abbildung unten lässt sich das erstellte User Interface sehen.

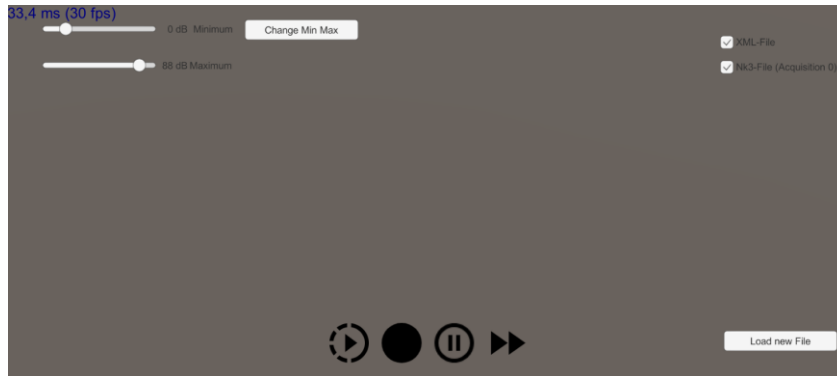


Abb. 51: User Interface

### 6.1 Rect-Transform

Rect-Transform ist eine neue Transform-Komponente, welche bei allen UI-Elementen anstatt der regulären Transform-Komponente verwendet wird. Diese Komponente hat als Attribute Position, Rotation und Skalierung genau wie ein normales Transform Objekt. Zusätzlich lassen sich jedoch auch Höhe und Breite einstellen, welche die Dimensionen eines Rechtecks festlegen.

Des Weiteren lässt sich der „Pivot-Point“ des Rect-Transforms konfigurieren. Die eingestellte Rotation und Position beziehen sich auf diesen Punkt. [21]

### 6.2 Canvas

Grundsätzlich musste zuerst ein Unity-Canvas erzeugt werden, an welches die einzelnen UI-Komponenten angehängt werden konnten. Wenn man ein UI-Element (unter „GameObject/Create UI“) erzeugt und noch kein Canvas vorhanden ist, wird dieses automatisch erzeugt und das Element angehängt. Obwohl es üblich ist, ein Canvas für alle UI-Elemente zu erzeugen, können auch mehrere erstellt werden. [22]

### 6.2.1 Canvas Scaler

Um das User Interface immer an die Größe des Bildschirms anzupassen, muss eine „Canvas Scaler“ Komponente hinzugefügt werden. Anfangs muss der „UI Scale Mode“ festgelegt werden. Dieser kann auf drei verschiedene Variablen gesetzt werden.

#### 6.2.1.1 Constant Pixel Size

Falls „Constant Pixel Size“ ausgewählt ist, werden die Positionen und Größen der einzelnen UI-Elemente anhand der Anzahl der Pixel am Bildschirm berechnet. Dies ist auch die Standardeinstellung, falls keine Canvas-Scaler Komponente vorhanden ist. Mit dem Skalierungsfaktor kann eine konstante Skalierung für alle UI Elemente festgelegt werden.

#### 6.2.1.2 Scale With Screen Size

Unter Verwendung des „Scale With Screen Size“ Modus wird die Größe und Position der einzelnen UI-Komponenten mithilfe einer festgelegten Referenzauflösung berechnet. Falls die momentane Auflösung größer als die Referenzauflösung ist, bleibt das Canvas bei der festgelegten Auflösung. Jedoch wird dann die Skalierung so erhöht, dass Größe und Positionen der Elemente passen. Ist die Bildschirmauflösung kleiner als die Referenzauflösung, wird die Skalierung so vermindert, dass das User Interface richtig ausgerichtet ist.

Diese Einstellung wird beim Projekt verwendet, da die Benutzeroberfläche so am besten ausgerichtet ist.

#### 6.2.1.3 Constant Physical Size

Bei Verwendung des „Constant Physical Size“ Modus, werden Position und Größe der UI-Elemente in physikalischen Größen, wie zum Beispiel Millimeter, angegeben.

## 6.3 Checkboxes zum Ein- und Ausblenden der Ansichten

Das Ein- und Ausblenden der verschiedenen Bauteilansichtsmöglichkeiten wurden mit Checkboxes umgesetzt. Pro Ansicht gibt es eine Checkbox, um diese ein- bzw. auszublenden.

### 6.3.1 Checkbox für Roboterprüfbahnen

Die Checkbox zum Ein- und Ausblenden der Roboterprüfbahnen wurde in Unity erstellt. Da jedes Mal, wenn die Applikation gestartet wird, die Prüflinien angezeigt werden, muss nicht auf das einzelne Bauteil im Code reagiert werden. In der Unity-Szene wurde mit Rechtsklick „UI/Toggle“ eine Checkbox erzeugt. Im Inspector wurden die Position und der Text des Kontrollkästchens

festgelegt. Weiters wurde ein „OnValueChanged“ Event an das Objekt angehängt, welches immer ausgelöst wird, wenn die Checkbox markiert bzw. nicht mehr markiert wird. Das Event wird im Inspector durch einen Klick auf das Plus-Icon erstellt. Danach kann festgelegt werden, welche Methode beim Auslösen des Events aufgerufen wird. Hierzu muss das Skript, in welchem sich die Methode befindet, per Drag&Drop in das Event gezogen und die gewünschte Funktion ausgewählt werden.

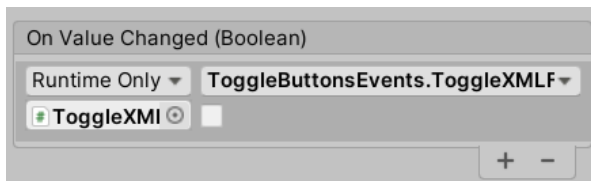


Abb. 52: Checkbox Event für Roboterprüfbahnen

Bei diesem Event wird die Methode „ToggleXMLFile“ aufgerufen. In der Methode selbst wird das Heatmap Skript mit der „GetComponent<Heatmap>()“ ermittelt. Da es sich beim Line Renderer, welcher für das Ein/Ausblenden benötigt wird, um eine globale Variable handelt, kann man auf ihn ganz normal referenzieren. Danach wird überprüft ob das „enabled“ Attribut des Renderers auf wahr gesetzt ist. Falls dies der Fall ist, wird das Attribut auf „false“ gesetzt, falls dies nicht stimmt wird das Attribut auf „true“ gesetzt.

```
System.Diagnostics.Debug.WriteLine("ToggleXMLFile " + xmlFileActive);
LineRenderer renderer = spawnSpheresObject.GetComponent<Heatmap>().renderer;
if (renderer.enabled)
{
    renderer.enabled = false; return;
}
renderer.enabled = true;
```

Abb. 53: Ein- und Ausblenden der Roboterprüfbahnen

### 6.3.2 Checkboxes für Heatmap

#### 6.3.2.1 Erzeugung der Checkboxes

Da es zur Erzeugung von mehr als einer Heatmap kommen kann, müssen auch mehrere Checkboxes zum Ein- bzw. Ausblenden der Heatmaps erstellt werden. Wie viele davon benötigt werden, wird bereits beim Einlesen des Ordners berechnet. Deswegen werden auch dort die Checkboxes generiert. In Unity wurde zuvor eine UI-Komponente erstellt und so angepasst, dass im Nachhinein nur noch die Position und der Text der Checkbox angepasst werden muss. Diese Komponente wurde als Prefab (siehe 2.1.3) abgespeichert und dem Skript per Drag&Drop als

globale Variable übergeben. Pro Heatmap wird mit der „Instantiate“ Methode eine Checkbox generiert. Als Namen wird unter anderem ein Index festgelegt. Dieser dient später zum Identifizieren der Heatmap, um zu wissen, welche ein- bzw. ausgeblendet werden soll. Des Weiteren wird die Position so eingestellt, dass der Abstand zu der vorherigen Checkbox jedes Mal exakt gleich ist.

```
GameObject nk3Toggle = Instantiate(toggleNk3Component, GameObject.Find("Canvas").transform);
nk3Toggle.name = "ToggleNk3File " + counter;
nk3Toggle.GetComponentInChildren<Text>().text = "Nk3-File (Acquisition " + counter + ")";
float newY = nk3Toggle.transform.position.y - counter * 26;
nk3Toggle.transform.position = new Vector3(nk3Toggle.transform.position.x, newY, 0);
```

Abb. 54: Erstellung der Checkboxes

### 6.3.2.2 Event

Ähnlich zur Roboterprüfbahnen-Checkbox muss auch der Heatmap-Checkbox ein Event angehängt werden. Dies wird wieder mit dem Plus-Icon im Inspector erreicht. Zusätzlich wird das Checkbox-Prefab als Parameter der Methode mitgegeben wird. Dies ist zur Identifizierung der ein- und auszubblendenden Heatmaps nötig.

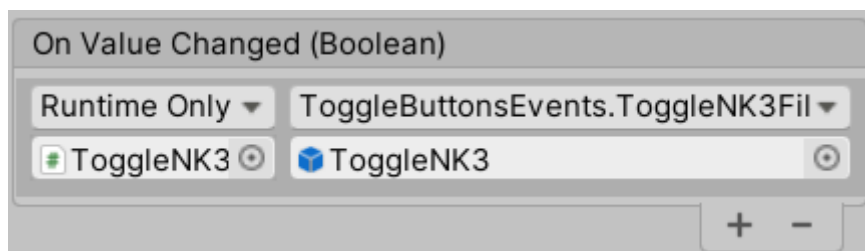


Abb. 55: Checkbox Event für Heatmap

In der aufgerufenen Methode wird zuerst der Index der Heatmap, unter Verwendung des Namens der Checkbox, bestimmt. Weiters wird das zugehörige Partikel System und die angehängte „Renderer“ Komponente ermittelt. Das „enabled“ Attribut wird dann entweder auf true oder falls gesetzt, je nach vorheriger Einstellung. War das Partikel System ausgeblendet und somit das Attribut auf false gesetzt, wird das Partikel System eingeblendet und somit das Attribut auf true gesetzt. Ist das Partikel System eingeblendet, geschieht es in umgekehrter Weise.



```
public void ToggleNK3File (GameObject checkbox)
{
    int numberHeatmap = int.Parse(checkbox.name.Split(' ')[1]);
    var particleSystemRenderer = GameObject.Find("SpawnSpheres").GetComponent<Heatmap>().allHeatmaps[numberHeatmap].GetComponent<Renderer>();
    if (particleSystemRenderer.enabled)
    {
        particleSystemRenderer.enabled = false;
    }
    else
    {
        particleSystemRenderer.enabled = true;
    }
}
```

Abb. 56: Ein- und Ausblenden der Heatmap

## 6.4 Einstellen von Minimum und Maximum

Für die Berechnung der Farben einer Heatmap wird ein Minimum und ein Maximum benötigt. Diese Werte sollen vom Benutzer im laufenden Programm eingestellt werden können. Dafür wurden zwei Schieberegler verwendet, einer für das Minimum und einer für das Maximum. Die Slider wurden in der Unity-Szene erstellt. Im Inspector wurden die Position und das Maximum bzw. Minimum beider Slider eingestellt. Ein zusätzlich registriertes „OnValueChanged“ Event ruft eine Methode auf, sobald sich der Wert des Sliders ändert.

### 6.4.1 Anzeigen der Schieberegler-Werte

Um das Einstellen des Maximums und Minimums benutzerfreundlicher zu gestalten, wird der aktuelle Wert eines Sliders immer in einem Textfeld angezeigt. Bei den Textfeldern handelt es sich um Text Komponenten aus der Unity UI Library. Für das Ändern der Texte wurde dem Slider das oben genannte Event hinzugefügt. In der Methode, welche aufgerufen wird sobald sich ein Wert der Slider ändert, wird die Anzeige auf den aktuellen Wert geändert.

### 6.4.2 Neuberechnung der Farben

Das Neuberechnen der Farben im Nachhinein ist nicht mehr möglich, weil die einzelnen Partikel nicht mehr zu identifizieren sind. So muss die gesamte Heatmap noch einmal mit den neuen Werten für das Minimum und Maximum generiert werden. Da dies sehr aufwändig ist, kann dies nicht bei jeder Wertänderung eines Sliders geschehen. Um trotzdem ein Ändern der Werte zu ermöglichen, wurde eine Schaltfläche hinzugefügt, welche bei Klick die neuen Werte ermittelt und die Methode zum Generieren der Heatmap aufruft.

Die neuen Werte werden ermittelt, indem alle Slider des Projekts auf ihren Namen überprüft werden. Wenn der Name des Sliders „SliderMin“ bzw. „SliderMax“ lautet wird das neue Minimum bzw. Maximum auf den Wert dieses Sliders gesetzt. Danach wird die „LoadNk3File“ Methode des Heatmap Skripts aufgerufen. In diesem werden alle Partikel aller Partikel Systeme gelöscht und die Heatmaps neu erstellt.

```
Heatmap heatmapScript = GameObject.Find("SpawnSpheres").GetComponent<Heatmap>();
Slider[] sliders = GameObject.FindObjectsOfType<Slider>();
foreach (Slider slider in sliders)
{
    if(slider.name == "SliderMin")
    {
        heatmapScript.minHeatmapValue = Convert.ToInt32(slider.value);
    }
    if(slider.name == "SliderMax")
    {
        heatmapScript.maxHeatmapValue = Convert.ToInt32(slider.value);
    }
}
heatmapScript.LoadNk3File(heatmapScript.pathToPosFile);
```

Abb. 57: Neuberechnung der Farben

## 6.5 Auswahl einer neuen Datei

Es soll möglich sein, in der laufenden Applikation ein neues Bauteil auszuwählen und anzeigen zu lassen. Dazu wurde ein Button im User Interface hinzugefügt, welcher im Inspector mit der passenden Position und dem richtigen Namen konfiguriert wurde. Wie auch schon bei dem Button, der für das Ändern von Minimum und Maximum der Heatmap zuständig ist, wurde hier ebenfalls ein Klick Event verwendet. Wenn auf die Schaltfläche gedrückt wird, öffnet sich wie beim ersten Start des Programmes der File Browser und ein neuer Ordner kann ausgewählt werden. Da beim einzelnen Erstellen jeder Bauteilansichten die vorherigen Daten gelöscht werden, muss nur die Methode, die auch beim Starten des Programmes verwendet wird, aufgerufen werden.

## 7 Integration der Test-Library

Martin Hinterberger, Software-Engineer bei Fill, sendete per Telegram eine Library, die zufällige Werte für die einzelnen Roboterachsen erstellt und returniert.

Die Werte schauen folgendermaßen aus:

```
RobotProperties = new FillStudioData
{
    A1 = A1,
    A2 = A2,
    A3 = A3,
    A4 = A4,
    A5 = A5,
    A6 = A6,
    XExt = xExt,
    Feedrate = feedrate
};
```

Abb. 58: Klasse RobotProperties

Die Werte kommen im Format TcpAxisPosition, welche im positiven oder negativen Bereich liegen können. Diese Daten werden im nächsten Schritt im JSON-Format gespeichert.

### 7.1 JSON – JavaScript Object Notation

JSON ist ein herkömmliches Datenformat mit einer breiten Applikations-Reichweite, welches auch als Ersatz für XML in AJAX Systemen dient.

JSON ist ein sprachenunabhängiges Datenformat. Es wird von JavaScript abgeleitet, aber die Mehrheit der modernen Programmiersprachen bieten Möglichkeiten, JSON-Code zu generieren und zu parsen. Der offizielle Internet Media Type für JSON ist „application/json“ und die Dateiendung „.json“. [23]

### 7.2 JSON Syntax

Das folgende Beispiel zeigt eine mögliche Syntax für JSON, welche die Werte für die Roboter abbildet.

## Recording-Funktion

JSON Beispiel:

```
{
  "RobotId": 3,
  "A1": 90.0,
  "A2": 56.36645,
  "A3": 76.53486,
  "A4": 0.0,
  "A5": 47.09869,
  "A6": 30.0,
  "XExt": 5273.7171,
  "Feedrate": 1.0,
  "ComponentName": "WINGLET_737_2mm",
  "ToolId": "PE_SLIDER"
}
```

Abb. 59: JSON TcpAxisPosition Beispiel

Die Auswahl des Dateiformats fiel deshalb auf JSON, da es an der Schule schon mehrmals benutzt wurde und eines der populärsten Formate ist.

## 8 Recording-Funktion

### 8.1 Recording-Event (Base-Klasse)

```
data = JsonConvert.DeserializeObject<TcpAxisPosition>(Encoding.UTF8.GetString(e.Body));
```

Abb. 60: JSON-Konvertierer

data	{Assets.Scripts.TcpAxisPosition}	Assets.Script..
A1	90	double
A2	0.00031	double
A3	120.0003	double
A4	0	double
A5	-20.00073	double
A6	0.00033	double
ComponentName	null	string
Feedrate	1	double
RobotId	2	int
ToolId	"TTU_TOOL"	string
XExt	0	double

Abb. 61: Veranschaulichung der data Variable

Die Variable data ist eine Instanz vom Datentyp TcpAxisPosition, die durch den JSON-Converter mit Werten befüllt wird.

```
if (RecordButton.isRecording == true)
{
    RobotsAxisDataReceived?.Invoke(this, new AxisPositionEventArgs(data));
}
```

Abb. 62: Recording-Kontrolle

Die IF-Verzweigung überprüft ob die globale Variable isRecording auf true gesetzt wurde. Ist die Schleife wahr, so wird das Event „RobotsAxisDataReceived“ mittels der Methode Invoke mit der Variable data aufgerufen.

## 8.2 Aufnahme-Button

```
public static bool isRecording;
public static DateTime dateTime;
public Button button;
public Sprite recordIcon;
public Sprite stopIcon;
// Start is called before the first frame update

0 references
void Start()
{
    isRecording = false;
}

0 references
public void clickedOnBtn()
{
    if (isRecording == false)
    {
        RecordingScript.Robots.Clear();

        button.image.sprite = stopIcon;
        isRecording = true;
        Debug.Log(isRecording);
        dateTime = DateTime.Now;
    }
    else
    if (isRecording == true)
    {
        button.image.sprite = recordIcon;
        isRecording = false;
        Debug.Log(isRecording);
    }
}
```

Abb. 63: RecordButton-Skript

Die globale Variable „isRecording“ wird in dem Skript „RecordButton“ initialisiert und behandelt. Dieses Skript überprüft ob der Aufnahme-Button gedrückt wurde. Falls dies zutrifft, so wird die globale Variable auf true gesetzt. Das Icon des Buttons wird zu einem Stopp-Icon (unten veranschaulicht) geändert und die globale Variable dateTime (welche ein Objekt eines DateTimes speichert) auf den jetzigen Zeitstempel gesetzt. Bei erneutem Drücken des Buttons wird die Variable wieder auf false gesetzt und das Icon ändert sich erneut zu einem Start-Icon.

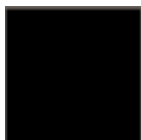


Abb. 64: Stopp Icon



Abb. 65: Start Icon

### 8.3 Aufnahme-Skript (RecordingScript)

```
0 references
void Start()
{
    Base.Instance.RobotsAxisDataReceived += AxisDataReceived;
}
```

Abb. 66: Aufnahme-Skript Start

Die Start-Methode wird aufgerufen, wenn das Skript zum ersten Mal verwendet wird. In dieser Methode wird dem EventHandler, mit dem zugewiesenen TcpAxisPosition Objekt aus der Base-Klasse, die Methode „AxisDataReceived()“ zugewiesen. Diese Methode wird jedem neuen aufgerufenen Objekt mitgegeben.

#### 8.3.1 AxisDataReceived

```
private void AxisDataReceived(object sender, AxisPositionEventArgs e)
{
    //e.Value.ComponentName = xmlData[0];
    json = JsonConvert.SerializeObject(e.Value);
    //string filename = $"{e.Value.ComponentName}_{e.Value.ToolId}.json";

    string filename = $"Test_{RecordButton.dateTime:yyyy_MM_dd_HH_mm_ss}.json";
    string path = $"{Assets/TestSavings/{filename}}";

    File.AppendAllLines(path, new List<string> { json });
}
```

Abb. 67: Methode AxisDataReceived

In dieser Methode wird das TcpAxisPosition Objekt, welches jetzt in den Parametern unter „AxisPositionEventArgs e“ zu finden ist, in die Datei geschrieben. Mit e.Value bekommt man das gesamte TcpAxisPosition Objekt. JsonConvert konvertiert Daten ins JSON-Format. Mit der Funktion „SerializeObject“ wird e.Value als Objekt serialisiert und in die Variable „json“ als String gespeichert. Filename und Path werden entsprechend gesetzt und mit File.AppendAllLines in die Datei geschrieben.

#### 8.3.2 XML-File

Kommentiert man den ersten Kommentar ein, dann wird der Name der Komponente, der aus einem XML-File genommen wird, überschrieben.

```
private void Awake()
{
    //xmlData = GetDataFromXml();
}
```

Abb. 68: Aufnahme-Skript Awake

Die Variable xmlData ist vom Datentyp String[], wobei sich an der nullten Stelle der Name der Komponente und an der ersten Stelle die ToolId befinden.

Die Initialisierung erfolgt in der Methode Awake(). Diese wird bei erstmaligem Laden des Skripts aufgerufen.

### 8.3.2.1 GetDataFromXML-Methode

```
private string[] GetDataFromXml()
{
    TextAsset textAsset = Resources.Load("Files/DRS_TTU_Winglet_OLP_2mm APPR_NEW") as TextAsset;

    //[0] = component name || [1] = tool id
    string[] result = new string[2];

    XmlDocument doc = new XmlDocument();
    doc.LoadXml(textAsset.text);

    //Get component name from xmlFile and add to array
    XmlNode nodeName = doc.DocumentElement.SelectSingleNode("/FillScanFile/Entry/Setup/Name");
    result[0] = nodeName.InnerText;

    //Get tool id from xmlFile and add to array
    XmlNode nodeTool = doc.DocumentElement.SelectSingleNode("/FillScanFile/Entry/Sequence/Channel/ToolID");
    result[1] = nodeTool.InnerText;

    return result;
}
```

Abb. 69: Methode GetDataFromXML

In der GetDataFromXml() Methode werden die Werte aus dem XML-File gelesen. Am Anfang erfolgt das Laden der Datei mit Hilfe des Befehls „Load“ aus dem TextAsset (siehe Punkt 0). Ein String[]-Objekt und eine XmlDocument-Klasse werden initialisiert. Diese Klasse repräsentiert ein XML-Dokument, die für das Laden, das Prüfen, das Editieren, das Hinzufügen und die Positionierung eines XML-Dokumentes zuständig ist. Mit LoadXml(Dateipfad) wird das Dokument aus dem angegebenen Pfad geladen. Für die Selektierung eines Knotens benötigt man die Klasse XML-Node, bei welcher man mit der Methode SelectSingleNode(path) den Knoteninhalt beschaffen kann. Der Pfad gibt an, unter welchen der durch „/“ getrennten Knoten sich der Inhalt befindet. Mit InnerText wird der tatsächliche String dem Array hinzugefügt und anschließend retourniert. Die Datei erstellt jetzt nach jedem Eintrag eine neue Zeile.

```
{ "RobotId":4, "A1":0.0, "A2":0.0, "A3":0.0, "A4":0.0, "A5":0.0, "A6":0.0, "XExt":0.0, "Feedrate":0.0, "ComponentName":"WINGLET_737_2mm", "ToolId":"FE_SLIDER" }
{ "RobotId":4, "A1":0.0, "A2":0.0, "A3":0.0, "A4":0.0, "A5":0.0, "A6":0.0, "XExt":0.0, "Feedrate":0.0, "ComponentName":"WINGLET_737_2mm", "ToolId":"FE_SLIDER" }
{ "RobotId":4, "A1":0.0, "A2":0.0, "A3":0.0, "A4":0.0, "A5":0.0, "A6":0.0, "XExt":0.0, "Feedrate":0.0, "ComponentName":"WINGLET_737_2mm", "ToolId":"FE_SLIDER" }
```

Abb. 70: Screenshot JSON-Datei

### 8.3.2.2 TextAssets

Sollen Dateien per Handy eingelesen und nachgefahren werden, ist dies auch möglich. Dazu verwendet man TextAssets anstatt eines Strings. Eine TextAsset-Klasse speichert Dateien, die mit verschiedenen Methoden ausgelesen werden können. Resources.Load() lädt die benötigte Datei, mit .text bekommt man die ganze Datei als String. Damit das Einlesen am Handy funktioniert, ist es wichtig, dass sich die Dateien in dem Ressourcenordner, den jedes Unity-Projekt haben muss, befinden. Beim Laden der Datei darf die Dateiendung nicht mitgegeben werden. [24]

```
TextAsset textAsset = Resources.Load("Files/DRS_TTU_Winglet_OLP_2mm APPR_NEW") as TextAsset;
```

Abb. 71: TextAsset laden eines XML-Files

```
doc.LoadXml(textAsset.text);
```

Abb. 72: XML-File laden



## 9 Datei einlesen (ReadData-Methode)

Diese Methode wird im Konstruktor des Base-Skripts aufgerufen. Sie liest eine Datei ein, welche anschließend konvertiert und in einer Liste gespeichert wird.

```
TextAsset asset = Resources.Load("Files/RectangleTestFile") as TextAsset;
```

Abb. 73: TextAsset laden eines Text-Files

```
private void ReadData()
{
    Console.WriteLine("*****ReadData*");
    Robots.Clear();

    //var linesTest = File.ReadAllLines(path);
    var assetLines = asset.text;
    string[] linesTest = assetLines.Split('\n');

    foreach (var line in linesTest)
    {
        Robots.Add(JsonConvert.DeserializeObject<TcpAxisPosition>(line));
    }
}
```

Abb. 74: Methode ReadData

Der erste Schritt besteht darin, die Robots-Liste zu leeren und ein TextAsset (siehe Punkt 0) zu erstellen. Darin muss der Pfad enthalten sein, um die Datei aufzufinden. Mit asset.text bekommt man den gesamten String der Datei, welcher dann auf ‚\n‘ (new Line -> neue Zeile) gesplittet wird. Mit der foreach-Schleife wird dann das Array, welches mit einzelnen Zeilen befüllt ist, durchiteriert, von JSON-Format auf TcpAxisPosition konvertiert und in der Robot-Liste gespeichert.

```
private readonly Timer timer;
```

Abb. 75: Erstellung Timer

```
private Base()
{
    //Setup_RabbitMQ();

    ReadData();

    timer = new Timer(50);
    timer.Elapsed += Timer_Elapsed;
    timer.Start();
}
```

Abb. 76: Timer Event zuweisen

Nachdem die Steuerung des realen Roboters alle 50 ms Werte liefert, eignet sich die Klasse Timer am besten. Es wird ein Objekt vom Typ Timer erstellt, dem man bei der Initialisierung das gewünschte Intervall setzen kann. Elapsed ist ein Event vom Timer, welches dann auftritt, wenn

## Pause Skript (PauseScript)

das gesetzte Intervall abgelaufen ist. Dem Event wird die Methode `Timer_Elapsed` angehängt, welche alle 50 ms aufgerufen wird. Mit `timer.Start()` startet man den Timer.

### 9.1 Timer\_Elapsed

```
private int idx = 0;
```

Abb. 77: Erstellung Index Variable

```
if (!PauseScript.isPause)
{
    // Debug.Log("*****" + Robots[idx].RobotId + Robots[idx].A1 + Robots[idx].A2);
    if (data.RobotId == 2)
    {
        Robot2AxisDataReceived?.Invoke(this, new AxisPositionEventArgs(data));
    }
    else
    {
        Robot1AxisDataReceived?.Invoke(this, new AxisPositionEventArgs(data));
    }
    idx++;
    Debug.Log(string.Format("Start of recording method"));
    if (RecordButton.isRecording == true)
    {
        RobotsAxisDataReceived?.Invoke(this, new AxisPositionEventArgs(data));
    }

    Debug.Log(string.Format("End of recording method " + RecordButton.isRecording));
}
```

Abb. 78: Methode `Timer_Elapsed`

Eine Variable Index wird mit dem Wert 0 erstellt, um als Index für die Roboter-Liste zu fungieren. In der IF-Verzweigung wird überprüft ob die globale Variable „isPause“ (siehe 10) false ist. Ist dies der Fall, so folgt eine zweite IF-Verzweigung, welche die RoboterId überprüft. Je nachdem welche ID gerade eingelesen wird, werden das zugehörige Roboterevent ausgelöst und die Daten als Parameter mitgegeben. Ist die Variable „isPause“ true, so wird die Roboter-Bewegung gestoppt. Für die dritte IF-Verzweigung siehe 8.1.

## 10 Pause Skript (PauseScript)

Das Skript wird dem Pause-Button zugewiesen. Die Icons (Play und Pause) werden im Inspector mitgegeben, um Fehler zu vermeiden.

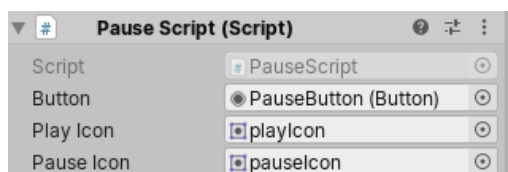


Abb. 79: Objekte an Pause-Skript zuweisen

## Pause Skript (PauseScript)

```
public class PauseScript : MonoBehaviour
{
    public static bool isPause = false;
    public Button button;
    public Sprite playIcon;
    public Sprite pauseIcon;

    0 references
    public void DoPauseToggle()
    {
        if (Time.timeScale > 0)
        {
            PauseMovement();
            button.image.sprite = playIcon;
        }
        else
        {
            UnPauseMovement();
            button.image.sprite = pauseIcon;
        }
    }

    1 reference
    public void PauseMovement()
    {
        isPause = true;
        Time.timeScale = 0;
    }

    1 reference
    private void UnPauseMovement()
    {
        isPause = false;
        Time.timeScale = 1;
    }
}
```

Abb. 80 Skript Pause

Um die Variablen im Unity-Inspektor nutzen zu können, muss das Schlüsselwort `public` verwendet werden. Im Pause-Skript werden der Button, die zwei Sprites (siehe 0) und die Variable „isPause“ benötigt. Letztere muss statisch sein, da sie auch im Base-Skript verwendet wird. In der Methode wird erstens kontrolliert, ob die Zeitskala größer als 0 ist. Der Standardwert der Zeitskala ist 1. Ist der Wert größer, so wird die Methode „PauseMovement()“ aufgerufen und das Play-Icon angezeigt. Trifft dies nicht zu, wird die Methode „UnPauseMovement ()“ aufgerufen.

### 10.1 DoPauseToggle()

Im Inspektor des Pause-Buttons wird ein Event durch einen Klick auf das Plus Icon angehängt. In diesem Event wird die Methode DoPauseToggle ausgewählt und somit bei jedem Knopfdruck aufgerufen.

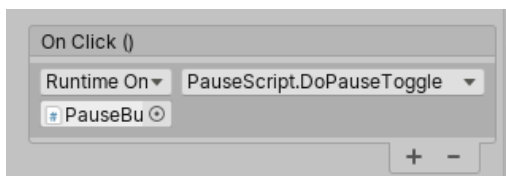


Abb. 81: Klick Event Pause

## 10.2 Roboterbewegung pausieren (PauseMovement)

In dieser Methode werden die Zeitskala auf 0 und „isPause“ auf true gesetzt, um die Methode ReadData() (siehe 9) zu pausieren.

## 10.3 Roboterbewegung starten (UnPauseMovement)

Hier werden die Zeitskala wieder auf 1 und „isPause“ auf false gesetzt, um die Methode ReadData() (siehe 9) zu starten.

# 11 Vuforia Basics

Vor Verwendung von Vuforia muss das Package von der Webseite [25] heruntergeladen werden.

Beim Importieren navigiert man in Unity auf Assets -> Import Package -> Custom Package und gibt den Pfad des Vuforia-Packages an. In einem Popup-Fenster wählt man aus, dass Vuforia zum scoped repository (siehe 11.1) hinzugefügt werden soll.

Nun kann man in Unity unter GameObject -> Vuforia Engine Vuforia Objekte erstellen, wie rechts im Bild zu sehen ist.

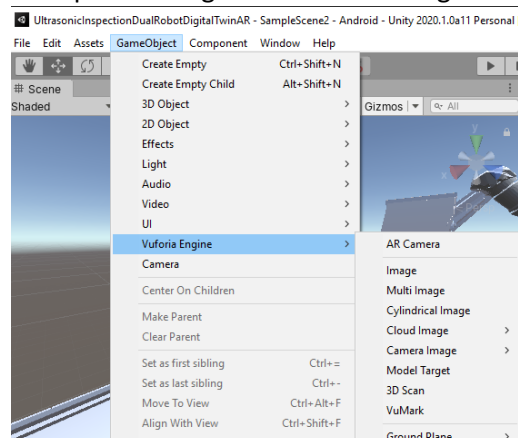


Abb. 82: Erfolgreich Vuforia importiert

## 11.1 Scoped repository

Dieses Repository speichert alle Packages, die verwendet werden. Durch den Gebrauch eines Repository wird sichergestellt, dass der Package-Manager (siehe 15.2.2) ein Paket immer nur demselben Repository zuordnet.

Setzt man zum Beispiel einen eigenen Server auf, indem man die Packages hostet, so könnte es sein, dass man ein Package von einem falschen Repository erhält. [26]

## 11.2 Vuforia-Account

Um Objekte mittels AR projizieren zu können, braucht man die AR-Kamera. Den dafür benötigten Aktivierungs-Key erhält man beim Erstellen eines Accounts auf der im Quellenverzeichnis angeführten Webseite. [27]

Nach der Anmeldung findet man in der Kategorie Develop -> License Manager den Button „Get Development Key“, mit welchem man einen eigenen kostenlosen Key erstellen kann.

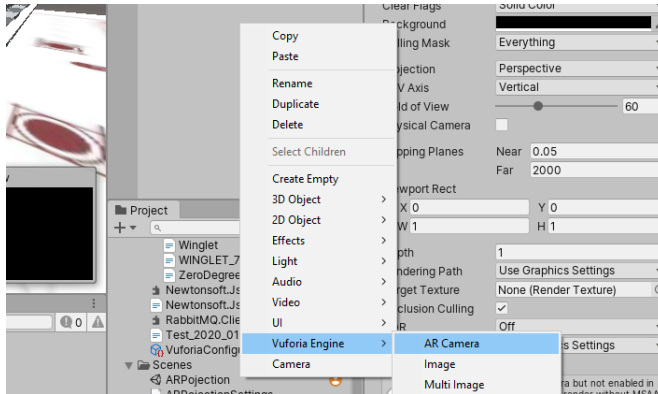


Abb. 83: Erstellung AR-Kamera

## 11.3 License-Key hinzufügen

In der Szene wird die AR-Kamera erstellt. Selektiert man diese, sieht man im Inspektor ein Vuforia Behaviour (Script), unter dem sich ein Button mit dem Namen „Open Vuforia Engine configuration“ befindet.

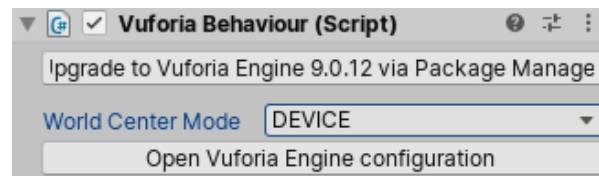


Abb. 84: Inspektor AR-Kamera

Beim Anklicken des Buttons öffnet sich ein neues Fenster im Inspektor, worin ein App License Key zu finden ist. In diesem Textfeld ist der generierte License Key einzufügen.

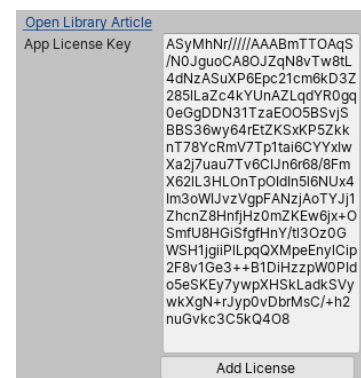


Abb. 85: Inspektor License Key

## 11.4 Vuforia Konfiguration

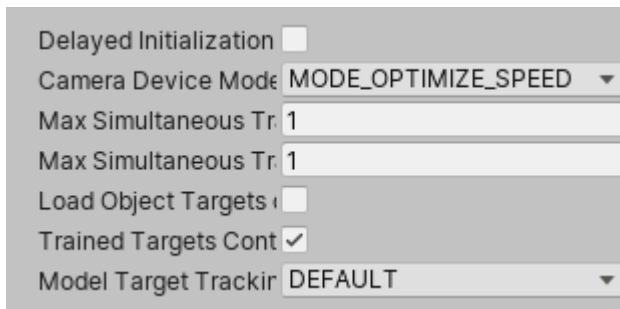


Abb. 86: Vuforia Konfiguration

Die Vuforia-Konfiguration befindet sich im selben Fenster wie der License Key.

### 11.4.1 Verzögerte Initialisierung

Vuforia kann nicht vor der Initialisierung gestartet werden. Wenn „Delayed Initialization“, ausgewählt wird, gibt es eine Verzögerung bei der Initialisierung. Diese muss manuell in der VuforiaRuntime-Klasse programmiert werden. Wenn die App nicht im AR-Modus gestartet wird, muss diese Funktion verwendet werden. Dieser Fall tritt zum Beispiel ein, wenn man im laufenden Programm per Buttonklick in den AR-Modus wechseln kann.

### 11.4.2 Gerätekamera Modus

Soll die Auflösung an die Kamera des Geräts angepasst werden, wählt man den Default Modus „MODE\_OPTIMIZE\_SPEED“ aus. Dieser Modus senkt die Auflösung (häufig 640 x 480), wodurch die Bildfrequenz steigt. Wählt man den Modus „MODE\_OPTIMIZE\_QUALITY“, so wird einerseits die Auflösung signifikant erhöht, andererseits steigt die Prozessorbelastung, was zu einer niedrigeren Bildfrequenz führen kann.

### 11.4.3 Vuforia-Datenbanken

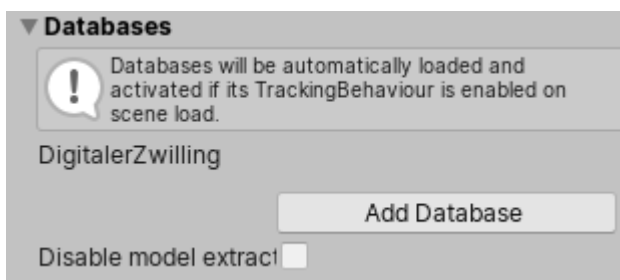


Abb. 87: Inspektor Datenbanken

Man kann die Datenbank jederzeit per Code befüllen bzw. leeren und aktivieren bzw. deaktivieren. Eine aktivierte, befüllte Datenbank ermöglicht der Vuforia Engine das Detektieren beliebiger Imagetargets, die in der Datenbank gespeichert sind.

### 11.4.3.1 Aufsetzen der Datenbank

Auf der Vuforia-Seite [27] sind zwei Manager vorhanden, einerseits der License Manager, welcher den License-Key enthält, und andererseits der Target Manager. Im Target Manager werden die Datenbanken angelegt und mit Imagetargets befüllt. Beim Anklicken des Buttons „Add Database“ öffnet sich ein Popup. Darin werden der Name und einer der drei Typen der Datenbank festgelegt. Der verwendete Device-Typ speichert die Imagetargets auf dem Gerät. Der Cloud-Typ speichert die Imagetargets auf der Cloud, die per http Connection abrufbar ist. Der dritte Typ, VuMark, ist eine neuartige Datenbank, in welcher man die Imagetargets individuell einrichten kann. Diese finden Großteils in Großunternehmen Gebrauch.

#### Add Target

Type:



File:

.jpg or .png (max file 2mb)

Width:

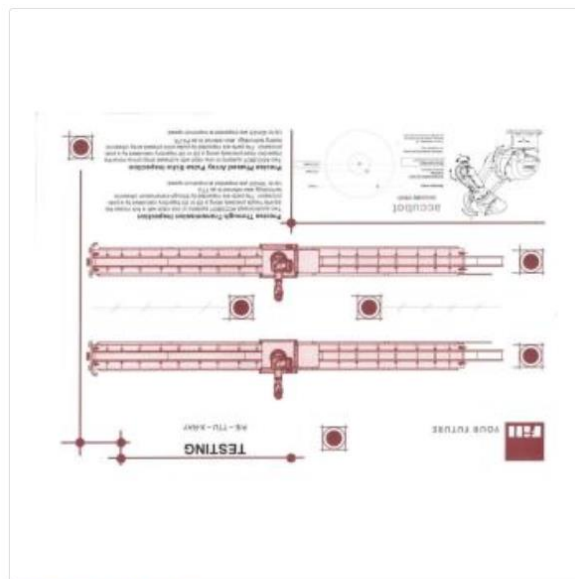
Enter the width of your target in scene units. The size of the target should be on the same scale as your augmented virtual content. Vuforia uses meters as the default unit scale. The target's height will be calculated when you upload your image.

Name:

Name must be unique to a database. When a target is detected in your application, this will be reported in the API.

**Abb. 88: Hinzufügen von Target**

Nach dem erfolgreichen Erstellen der Datenbank, kann mit dem Button „Add Target“ ein spezifisches Imagetarget hinzugefügt werden. Im nächsten Schritt wird der Typ des Imagetargets gewählt. Das gewünschte Bild wird ausgewählt, die Größe und der Name des Bildes werden angegeben. Zum Speichern der Änderungen muss der Button „Add“ geklickt werden.



Update Target Show Features

Type: Single Image

Status: Active

Target ID: 0d18c71c4f374aa8be046cf0f1fa0b5e

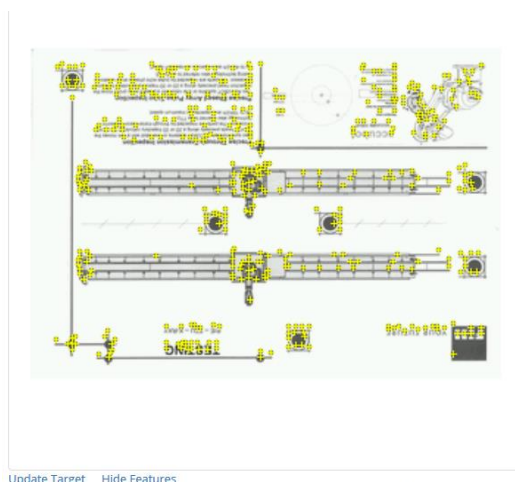
Augmentable: ★★★★★

Added: Feb 10, 2020 20:46

Modified: Feb 10, 2020 20:46

**Abb. 89: Imagetarget Informationen**

Rechts unten im Bild sind Informationen über das Imagetarget aufgelistet. Die wichtigste Information ist die Anzahl an Sternen. Sie geben an, wie gut das Bild von Vuforia erkannt wird.



Update Target Hide Features

**Abb. 90: Imagetarget mit Features**



Mit „Show Features“ werden für Vuforia erkennbare Punkte angezeigt. Je mehr Features, desto besser und höher ist die Erkennungsrate.

Bei einem passenden Imagetarget kann per „Download Database“ die Datenbank heruntergeladen und in Unity importiert werden. Es öffnet sich ein Popup, in welchem ausgewählt werden muss, auf welcher Plattform diese Datenbank verwendet werden soll. In diesem Fall muss der Unity Editor gewählt werden. Die Datei wird dann, wie das Vuforia-Package (siehe 11), in Unity importiert.

## 12 AR Projektion (Imagetarget und AR Kamera)

### 12.1 AR Kamera



Abb. 91: Abbildung von AR-Kamera

Nachdem die Datenbank importiert wurde, kann das in der Datenbank hinzugefügte Bild in der Szene eingefügt werden. Wie oben in der Abbildung ersichtlich, wird das Objekt „ARCamera“ benötigt. Durch einen Rechtsklick in der Szene öffnet sich ein Kontextmenü. Unter VuforiaEngine -> AR Camera wird eine auf das Objekt gerichtete Kamera erstellt.

### 12.2 Imagetarget

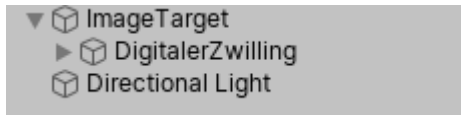


Abb. 92: Abbildung von Imagetarget

Durch einen Rechtsklick in der Szene öffnet sich ein Kontextmenü. Unter VuforiaEngine -> ImageTarget wird das Imagetarget erstellt und das Roboter-Prefab (siehe 2.1.3) in Unity „DigitalerZwilling“ hierarchisch untergeordnet angehängt.

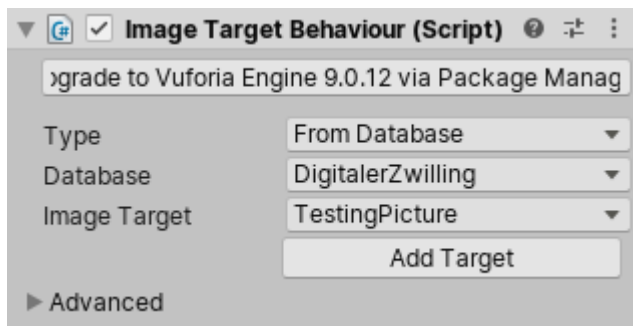


Abb. 93: Inspektor Imagetarget

Im Inspektor des Imagetargets wird bei Type „From Database“ oder „From Image“ gewählt. Bei Ersterem wird das Imagetarget automatisch aus der Datenbank entnommen, andernfalls muss es manuell zugewiesen werden.

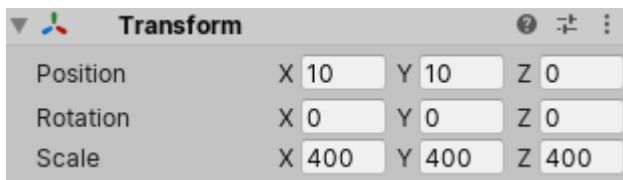


Abb. 94: Skalierung Imagetarget

Achtung! Ist die Größe, die beim Erstellen des Image-Targets gesetzt wurde, zu klein, besteht die Möglichkeit, dass das Bild in Unity nicht mehr sichtbar ist. In diesem Fall muss die Skalierung im Inspector erhöht werden.

Nun ist die Projektion im eigentlichen Sinne fertig. Beim Öffnen der App wird unerwünschter Weise das GUI zeitlich vor der Roboterprojektion angezeigt. Um dies zu verhindern, musste dem Imagetarget das ButtonPopup-Skript (siehe 12.2.1) zugewiesen werden.

### 12.2.1 Buttons mit AR anzeigen (ButtonPopup Skript)

```
private TrackableBehaviour mTrackableBehaviour;  
  
public GameObject canvas;  
  
0 references  
void Start()  
{  
    mTrackableBehaviour = GetComponent<TrackableBehaviour>();  
    if (mTrackableBehaviour)  
    {  
        mTrackableBehaviour.RegisterTrackableEventHandler(this);  
    }  
}  
  
0 references  
public void OnTrackableStateChanged(  
    TrackableBehaviour.Status previousStatus,  
    TrackableBehaviour.Status newStatus)  
{  
    if (newStatus == TrackableBehaviour.Status.DETECTED ||  
        newStatus == TrackableBehaviour.Status.TRACKED)  
    {  
        TargetFound();  
    }  
}  
  
1 reference  
private void TargetFound()  
{  
    canvas.SetActive(true);  
}
```

Abb. 95: ButtonPopup-Skript

Es werden zwei globale Variablen, TrackableBehaviour und GameObject, angelegt. Das GameObject „canvas“ wird per Instruktor zugewiesen.

Zum Verständnis der TrackableBehaviour-Klasse siehe 12.2.1.1. In der Start() Methode wird als erstes die Variable „mTrackableBehaviour“ mit GetComponent initialisiert. Anschließend wird in der IF-Verzweigung abgefragt, ob eine TrackableBehaviour-Komponente gefunden wurde. Trifft dies zu, werden alle TrackableBehaviours benachrichtigt, und die vordefinierte Methode „OnTrackableStateChanged()“ aufgerufen. Darin wird über die Variable newStatus überprüft, ob

das Target gefunden wurde. Ist dies der Fall, werden die Buttons eingeblendet. Ändert sich der Status nicht mehr, bleibt das Default- Canvas inaktiv.

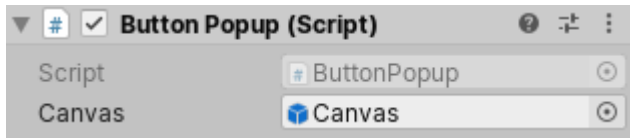


Abb. 96: Canvas über Inspektor an ButtonPopup-Skript zuweisen

#### 12.2.1.1 TrackableBehaviour

Diese Klasse dient als Base-Klasse für alle „TrackableBehaviours“ in Vuforia. Sie stellt Funktionen bereit, die nötig sind, um das Target detektieren zu können.

Der Status ist ein Enum und beinhaltet fünf Zahlen, welche unten in der Tabelle erklärt sind. Trackable ist jenes Imagetarget, das benötigt wird, um ein Objekt projizieren zu können. [28]

Enumerator
------------

NO_POSE = 0	Es wurden keine Features (siehe 11.4.3.1) des Trackables erkannt.
LIMITED = 1	Das Tracking funktioniert nur eingeschränkt.
DETECTED = 2	Das Trackable wurde grundsätzlich erkannt.
TRACKED = 3	Das Trackable ist aktuell erkennbar.
EXTENDED_TRACKED = 4	Das Trackable wurde in erweiterter Form gefunden.

## 13 Android-APK Erstellen

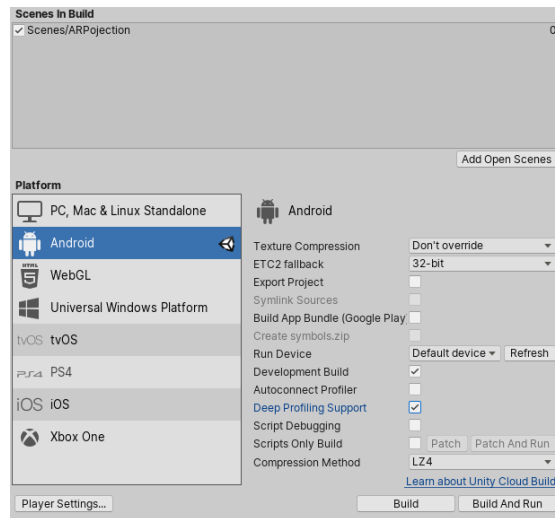


Abb. 97: Abbildung der Build Einstellungen

Unter File -> Build Settings -> Android wird aus der ausgewählten Szene eine Applikation erstellt. Mit dem Button „Build“ wird die oben aufgelistete Szene auf das, per USB-Kabel angeschlossene Handy, in Form einer APK installiert.

So wird grundsätzlich eine APK erstellt. Ist jedoch die Umgebungsvariable „JAVA\_HOME“ nicht gesetzt, wird folgende Fehlermeldung die Erstellung abbrechen:

*Gradle failed to fetch dependencies. ... ERROR: JAVA\_HOME is not set and no 'java' command could be found in your PATH.*

Abb. 98: Error: JAVA\_HOME

### 13.1 JAVA\_HOME setzen

Um dieses Problem zu beseitigen, muss unter Windows -> Systemumgebungsvariablen bearbeiten -> Umgebungsvariablen die Umgebungsvariable „JAVA\_HOME“ gesetzt werden. Mit einem Buttonklick auf „Neu...“ erstellt man eine neue Variable. Es öffnet sich ein Popup, in welchem der Name der Variable und der Wert mittels „Verzeichnis durchsuchen...“, gesetzt werden. Der Pfad des installierten JDKs (siehe 13.1.1) dient als Wert der neu erstellten „JAVA\_HOME“ Variable.

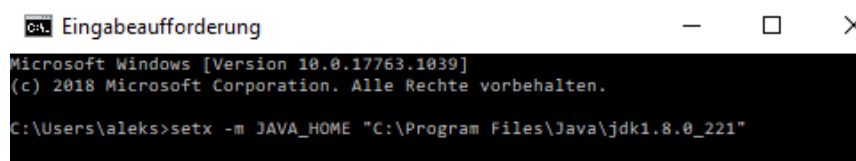


Abb. 99: Befehl in Eingabeaufforderung

Man kann die Variable permanent setzen, indem man die Eingabeaufforderung öffnet und die Variable, mit dem oben abgebildeten Befehl, fixiert. [29]

### 13.1.1 Java Development Kit (JDK)

Das Java Development Kit wird zur Entwicklung von Java-Programmen eingesetzt. Ohne einer JDK-Version ist die Entwicklung nicht möglich. Das JDK umfasst die Java-Laufzeitumgebung, den Java-Compiler und die Java-APIs. Auf der Oracle-Hauptseite ist die Installationsdatei auffindbar. [30]

## 14 Performance

Unter Build Settings -> Player Settings können die Eigenschaften der App modifiziert werden.

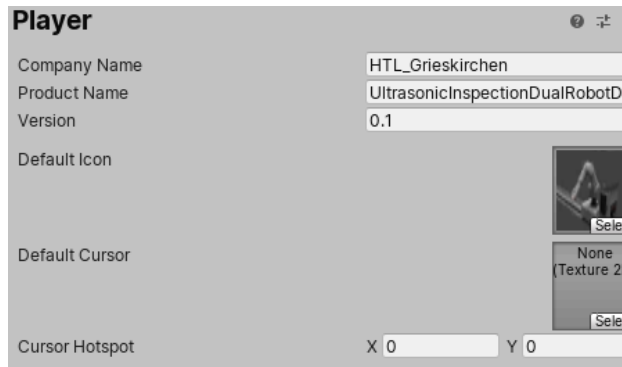


Abb. 100: Abbildung der Player Einstellungen

Hier werden der Produktname, der Name der Ersteller, die Version und das App-Icon der App gesetzt.



Abb. 101: Abbildung der DPI

Durch das Einstellen der Fixed DPI wurde die Performance drastisch verbessert. Die Qualität wurde in akzeptablem Ausmaß niedriger und die Bildfrequenz stieg von 5-7 auf 15-20 FPS, wodurch das Ruckeln deutlich verbessert wurde.



Abb. 102: Abbildung der Stack Trace

Die „Stack Trace“ sollten ausgeschaltet werden, da diese etwaige Fehlermeldungen oder Warnungen ausgeben, was zu Performance-Problemen führen kann.

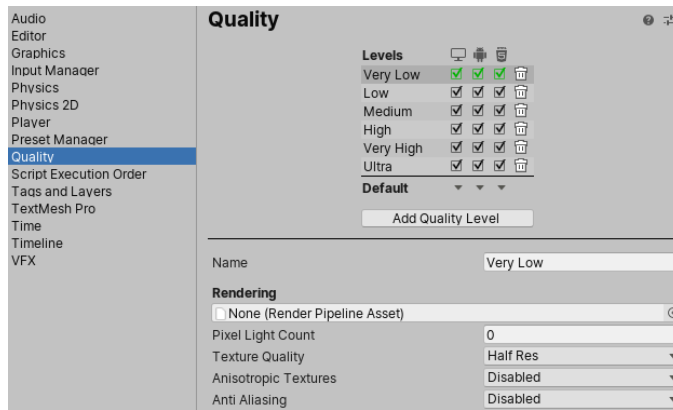


Abb. 103: Abbildung der Bildqualität

Unter der Registerkarte „Quality“ wird die Bildqualität auf „Very Low“ gesetzt.

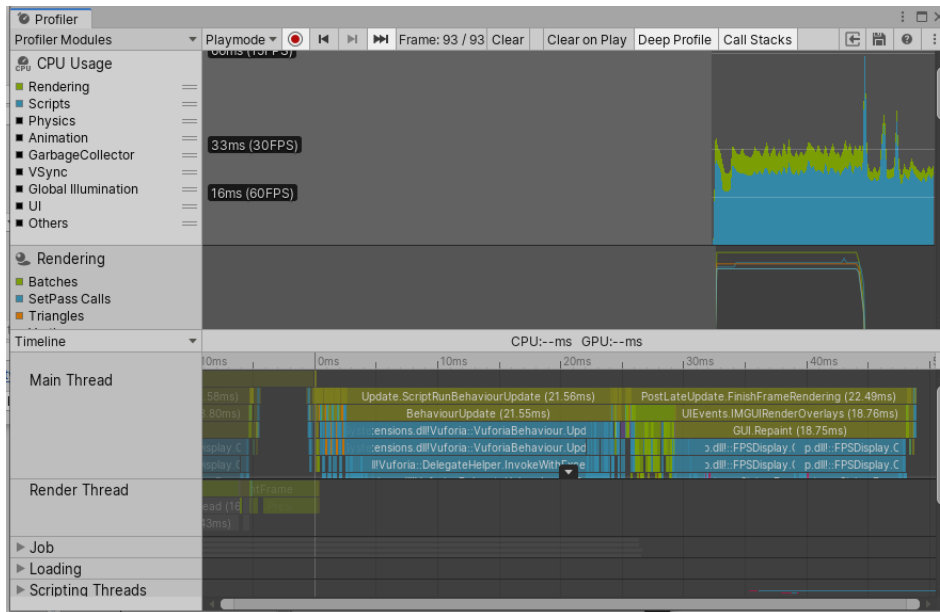
Dies ist nötig, da Handys nicht so leistungstark sind. Aufgrund des kleinen Handybildschirms wird das Endergebnis jedoch nicht dramatisch beeinflusst.

### 14.1 Unity Profiler

Der Unity Profiler wurde verwendet, um die Ursache der niedrigen Bildfrequenz ausfindig zu machen. Dieses Tool liefert Performanceinformationen über die laufende Applikation. Er kann mit Endgeräten, wie zum Beispiel einem Handy, verbunden werden, um die Performance der Applikation auf diesem Gerät zu testen. Daten aus den Bereichen CPU, Speicher und Audio werden angezeigt. Es ist ein hilfreiches Tool, um die performanceschwachen Bereiche ausfindig zu machen. Die Ergebnisse werden in Graphen veranschaulicht, um die Zu- und Abnahme der FPS-Anzahl erkennbar zu machen. Der Unity Profiler ist unter Window-> Analysis -> Profiler zu finden. [31]



## Performance



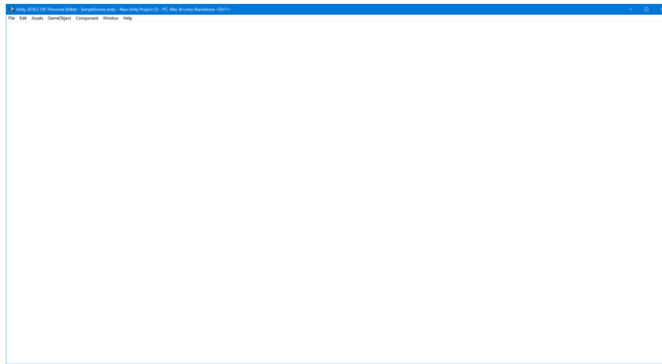
**Abb. 104: Unity Profiler**

Auf dem Bild erkennt man Ausreißer, welche unter 10 FPS fallen. Dies tritt auf, wenn Vuforia das Imagetarget entdeckt und die Roboter darauf projiziert. Daraus wurde ersichtlich, dass die niedrige FPS-Anzahl aufgrund der Roboterprojektion entsteht. Aus diesem Grund wurde ein eigenes Roboter-Modell (siehe 3.2.4) für die Projektion am Handy erstellt.

## 15 Unity Probleme

### 15.1 Unity rendert nicht

Ein großes Problem zu Beginn war, dass Unity beim Startvorgang nicht gerendert werden konnte. Es war nur ein weißes Fenster auf dem Bildschirm zu sehen, wie im Screenshot erkennbar.



**Abb. 105: Error: Unity rendert nicht**

Herkömmlich wird dieses Problem durch das Starten von Unity mit der Intel Grafikkarte gelöst [32]. Im benutzten Laptop war jedoch nur eine NVIDIA-Grafikkarte verbaut. Nicht die Grafikkarte selbst war jedoch das Problem, sondern Konflikte zwischen Unity und dem Treiber der Grafikkarte. Mit der Veröffentlichung der Unity-Version 2020.1.0a11 wurde das Problem behoben.

### 15.2 Unity-Versionen

Das Arbeiten mit Unity lief lange Zeit problemlos. Aus ungeklärter Ursache funktionierte das Programm von einem Tag auf den anderen auf einem Laptop jedoch nicht mehr. Erneut wurde viel Zeit in Recherchen investiert, doch das Problem konnte nicht gelöst werden. Die Version auf allen Laptops zu wechseln war keine Lösung, da dann das Rendern auf einem Gerät wieder nicht funktionierte (siehe 0). Schließlich wurde nur auf dem Laptop, auf welchem sich Unity nicht mehr hatte starten lassen, die neueste Unity-Version installiert.

Ausgenommen des Veröffentlichens neuer Änderungen war alles funktionsfähig. Einige Fehlermeldungen über ein undefiniertes Package wurden ausgegeben.

## Unity Probleme

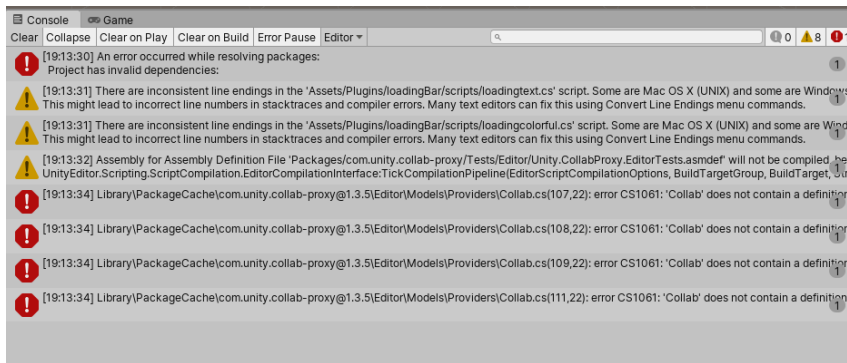


Abb. 106: Error: Package collab-proxy undefiniert

Diese Fehlermeldungen entstanden durch die neue Unity-Version. Das Package „collab-proxy“ muss am neusten Stand sein, was in älteren Versionen nicht gegeben ist. So musste im Explorer der Packages-Ordner geöffnet werden. Darin befindet sich eine Datei namens „manifest.json“ (siehe 15.2.1), worin „collab-proxy“ manuell wieder an die niedrigere Version angepasst werden muss. [33]

Alt

```
"dependencies": {  
  "com.ptc.vuforia.engine": "8.6.10",  
  "com.unity.collab-proxy": "1.2.15",  
}
```

Abb. 108: Package collab-proxy alt

Neu

```
"dependencies": {  
  "com.ptc.vuforia.engine": "8.6.10",  
  "com.unity.collab-proxy": "1.3.5",  
}
```

Abb. 107: Package collab-proxy neu

### 15.2.1 Manifest.json

Beim Laden eines Unity-Projektes liest der Unity Package-Manager (siehe 15.2.2) die Datei „manifest.json“ ein. Mithilfe dieser Datei ermittelt der Package-Manager, welche Packages geladen werden müssen. Packages, die installiert bzw. deinstalliert werden, sind im Manifest auffindbar. Diese Datei arbeitet mit einer Liste vom Typ „Dependencies“ (siehe 15.2.3) [34].

### 15.2.2 Package-Manager

Ein Package ist ein Container, welcher verschiedene Erweiterungen oder Assets speichert, wie zum Beispiel:

- Editor tools und Libraries
- Runtime tools und Graphics pipeline
- Animationen und Textures

Diese werden einheitlich auf einen Blick im Package-Manager angezeigt, welcher in Window -> Package Manager gefunden wird. Beim Öffnen eines Projektes erfolgt als Erstes das Einlesen der Datei „manifest.json“ (siehe 15.2.1) durch den Package-Manager, um herauszufinden, welche Packages geladen werden sollten und welche nicht. Weiters sendet er für jedes benötigte Package einen Request zum Registry-Server, welcher alle zur Verfügung stehenden Packages beinhaltet. Der Server sendet die entsprechenden Informationen und Daten an den Package-Manager zurück. Jedes Projekt besitzt eine eigene Manifest-Datei, welche als „dependencies“ (siehe 15.2.3) im Projekt abgespeichert wird. Der Package-Manager verwendet für die Kommunikation mit Usern, Manifests und Registries drei Interfaces. [35]

### 15.2.2.1 User-Interface

Dieses Fenster wird verwendet, um unmittelbar nach Features zu suchen. Außerdem ermöglicht es problemlos, Packages nach Bedarf zu installieren bzw. zu aktualisieren und Konflikte bei den „dependencies“ zu lösen. [35]

### 15.2.2.2 Package-Manifest

Der Package Manager stellt einen Unity-Inspektor zur Verfügung, welcher Informationen eines Packages darstellt. Zusätzlich können dank diesem Feature auch die Informationen eines spezifischen Packages direkt in Unity bearbeitet werden. [35]

### 15.2.2.3 Scripting API

Dieses Interface ermöglicht dem User die Interaktion mit dem Package-Manager per Code. Es wird verwendet, wenn Packages nach bestimmten Kriterien sortiert, gelöscht oder neu installiert werden müssen. [35]

## 15.2.3 Dependencies-Objekt

Das „dependencies“ Objekt gibt eine Liste der verfügbaren Packages an. Dieser Eintrag im Manifest listet die Namen der Packages und die für das Projekt benötigte Version auf. Die Liste beinhaltet jedoch nicht die indirekten Dependencies. Dies sind Packages, welche von anderen Packages benötigt werden. [36]

## 15.3 Problem bei AR-Kamera

Bei der Verwendung des „AR Camera“ Objektes in der Unity-Szene wird im Tab „Scene“ eine schwarze Fläche angezeigt. Nach dem Starten der Unity-Szene werden aber die Objekte angezeigt und die Kamera funktioniert einwandfrei. Dieses Problem wurde durch das Zurücksetzen der OpenGL Library Version von 3 auf 2.1 (siehe 15.4) gelöst [37]. Unter dem Tab Edit -> Project Settings -> Other Settings kann die Version der OpenGL Library zurückgesetzt werden.

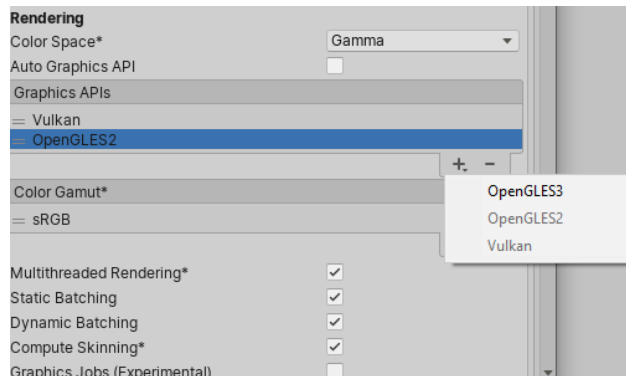


Abb. 109: OpenGL zurücksetzen

Mit dem Plus im Inspektor (siehe Abbildung 106 „OpenGL zurücksetzen“) wird die Version auf 2.0 zurückgesetzt.

## 15.4 Open Graphics Library (OpenGL)

Für die Entwicklung von 2D - und 3D-Apps kann die plattform- und programmiersprachenübergreifende „OpenGL“ Library verwendet werden. Diese Library führt zu einer Interaktion mit der „graphics processing unit“ (GPU), um ein schnelleres Rendern zu gewährleisten. [38]

### 15.4.1 Unterschied zwischen OpenGL 2.1 und OpenGL 3.0

OpenGL 3 stellt, im Gegensatz zu OpenGL 2.1, weitere Komponenten, wie zum Beispiel Funktionen, Frame Buffer, Texture Arrays und Rendering zur Verfügung. Für das Programmieren von Spielen eignet sich die neue Version aufgrund der hinzugefügten Komponenten besonders. Jedoch wird OpenGL 3.0 nicht von jeder Library unterstützt, wodurch Konflikte mit anderen Komponenten entstehen können. [39]

## 16      **Danksagung**

Hiermit möchten wir uns herzlich bei unserem Betreuungslehrer DI Alfred Doppler und bei unserem Kooperationspartner, der Firma Fill GmbH mit Sitz in Gurten bedanken. Besonderer Dank gilt Herrn Gramberger, Herrn Murauer und Herrn Hinterberger, die unsere Ansprechpartner bei der Firma Fill waren.

## 17      **Statement des Auftraggebers**

Fill ist ein international führendes Maschinen- und Anlagenbau-Unternehmen für verschiedenste Industriebereiche. Modernste Technik und Methoden in Produktion, Kommunikation und Management zeichnen das Familienunternehmen aus. Die Geschäftstätigkeit umfasst die Bereiche Metall, Kunststoff und Holz für die Automobil-, Luftfahrt-, Sport- und Bauindustrie.

In der Aluminium-Entkerntechnologie, in der Gießereitechnik, in der Holzbandsägetechnologie sowie für Ski- und Snowboardproduktionsmaschinen ist das Unternehmen Weltmarkt- und Innovationsführer.

Im Bereich von NDT (non-destructive-testing) konnte in den letzten Jahren mit hochspezialisierten Maschinen in Verbindung mit intelligenter Software ein Zeichen im Markt gesetzt werden.

Fill entwickelt und liefert hierbei Turnkey Prüfsysteme für die großen Luftfahrtkonzerne, für große Strukturbauteile von zivilen Flugzeugen aus Faserverbundstoffen (hauptsächlich CFRP).

Im Zuge des gegenständlichen Projektes entwickelten die Projektanden ein reales 1:1 Abbild einer dieser Referenzanlagen von Fill. Die CAD Modelle wurden entsprechend angepasst, optimiert und letztlich auf Basis von Unity kinematisiert. Die daraus resultierende Simulation wurde mit dem sogenannten Digitalen Zwilling der Anlagensteuerung verbunden. Hierbei wurde MQTT als Protokoll für die Schnittstelle des Organisations- und Planungstools "FILL.Studio" verwendet, um die Achswerte der NC-Steuerung abzugreifen. Die Entwicklung einer "Recording" Funktion soll künftig dem Vertriebsteam von Fill helfen, bei Kundenterminen realitätsgetreue Simulationen von entsprechenden Prüfsequenzen vorführen zu können. Die Funktionalität, ein Scanresultat direkt im Prüfbereich am Echtbauteil anzuzeigen ist ein weiterer Schritt in Richtung Digitalisierung der NDT Welt. Hierbei wurde das binäre \*.NK3 Format entsprechend ausgelesen, ein bewährter Standard der Luftfahrtindustrie für die Speicherung von Ultraschallprüfdaten ohne Verlust der Zuordnung zur 3D Geometrie der geprüften Bauteile. Über die Schnittstelle zu FILL.Studio werden auch Informationen zum aktuell am Roboter befindlichen EOAT (End Of Arm Tool) übertragen, und entsprechend visualisiert.

Das technische Wissen sowie die schnelle Auffassungsgabe der Projektanden bildeten die perfekte Basis für die äußerst erfolgreiche Umsetzung dieses zukunftsweisenden Projektes.

*Stefan Murauer, Thomas Gramberger*

## Abbildungsverzeichnis

Abb. 1: Prefabs .....	5
Abb. 2: Sprite extrahiert.....	6
Abb. 3: Sprite Inspektor .....	7
Abb. 4: Versionsgeschichte .....	8
Abb. 5: Unity commit .....	8
Abb. 6: Einbindung Visual Studio .....	9
Abb. 7: Robotermodell.....	14
Abb. 8: Gesamtmodell .....	14
Abb. 9: Material Roboter Grau.....	15
Abb. 10: Material Alu .....	15
Abb. 11: Material Gummi.....	16
Abb. 12: Material Kupfer.....	16
Abb. 13: Material Glas.....	16
Abb. 14: Material Scharniere .....	17
Abb. 15: Material Fill Logo .....	17
Abb. 16: Rig Detailansicht .....	18
Abb. 17: Klasse TcpAxisPosition .....	20
Abb. 18: Methode SetUp_RabbitMQ 1 .....	21
Abb. 19: Methode SetUp_RabbitMQ 2 .....	21
Abb. 20: Methode Consumer_Received .....	21
Abb. 21: RabbitMqReceiverMovement Start.....	22
Abb. 22: Methode AxisDataReceived.....	22
Abb. 23: Methode MoveFancy.....	23
Abb. 24: AxisDataReceived Achse 1 .....	24
Abb. 25: AxisDataReceived Achse 2 .....	25
Abb. 26: AxisDataReceived Achse 3 .....	25
Abb. 27: AxisDataReceived Achse 6 .....	25
Abb. 28: Methode Rotate .....	26
Abb. 29: Methode getRenderers .....	26
Abb. 30: Methode CheckTool.....	27
Abb. 31: RabbitMqReceiverRotations Achse 6 Update .....	27
Abb. 32: Ausschnitt der Binärdatei .....	28
Abb. 33: Einlesen der Positionen .....	29
Abb. 34: Einlesen der Fehlerwerte.....	30



Abb. 35: Farbverlauf [33] .....	31
Abb. 36: Berechnung Farbe .....	31
Abb. 37: Berechnung bei Farbwert über 75% .....	31
Abb. 38: Berechnung bei Farbwert zwischen 50% und 70% .....	32
Abb. 39: Berechnung bei Farbwert zwischen 25% und 50% .....	32
Abb. 40: Berechnung bei Farbwert unter 25% .....	32
Abb. 41: Partikel Erstellung .....	33
Abb. 42: Erstellung Partikelsystem .....	34
Abb. 43: Ausschnitt XML-Datei .....	35
Abb. 44: Einlesen der Prüfbahnen .....	36
Abb. 45: Erzeugen der Prüfbahnen .....	37
Abb. 46: Initialisierung des Blender Modells .....	37
Abb. 47: Unity Asset: File Browser .....	38
Abb. 48: Importieren des File Browsers .....	38
Abb. 49: Automatischer Start des File Browsers .....	39
Abb. 50: Automatische Auswahl des NK3-Ordners .....	40
Abb. 51: User Interface .....	41
Abb. 52: Checkbox Event für Roboterprüfbahnen .....	43
Abb. 53: Ein- und Ausblenden der Roboterprüfbahnen .....	43
Abb. 54: Erstellung der Checkboxes .....	44
Abb. 55: Checkbox Event für Heatmap .....	44
Abb. 56: Ein- und Ausblenden der Heatmap .....	45
Abb. 57: Neuberechnung der Farben .....	46
Abb. 58: Klasse RobotProperties .....	47
Abb. 59: JSON TcpAxisPosition Beispiel .....	48
Abb. 60: JSON-Konvertierer .....	48
Abb. 61: Veranschaulichung der data Variable .....	48
Abb. 62: Recording-Kontrolle .....	49
Abb. 63: RecordButton-Skript .....	49
Abb. 64: Stopp Icon .....	49
Abb. 65: Start Icon .....	49
Abb. 66: Aufnahme-Skript Start .....	50
Abb. 67: Methode AxisDataReceived .....	50
Abb. 68: Aufnahme-Skript Awake .....	50
Abb. 69: Methode GetDataFromXML .....	51
Abb. 70: Screenshot JSON-Datei .....	51

Abb. 71: TextAsset laden eines XML-Files.....	52
Abb. 72: XML-File laden .....	52
Abb. 73: TextAsset laden eines Text-Files.....	53
Abb. 74: Methode ReadData .....	53
Abb. 75: Erstellung Timer.....	53
Abb. 76: Timer Event zuweisen.....	53
Abb. 77: Erstellung Index Variable .....	54
Abb. 78: Methode Timer_Elapsed .....	54
Abb. 79: Objekte an Pause-Skript zuweisen .....	54
Abb. 80 Skript Pause .....	55
Abb. 81: Klick Event Pause .....	55
Abb. 82: Erfolgreich Vuforia importiert .....	56
Abb. 83: Erstellung AR-Kamera .....	57
Abb. 84: Inspektor AR-Kamera.....	57
Abb. 85: Inspektor License Key .....	57
Abb. 86: Vuforia Konfiguration .....	58
Abb. 87: Inspektor Datenbanken .....	58
Abb. 88: Hinzufügen von Target.....	59
Abb. 89: Imagetarget Informationen .....	60
Abb. 90: Imagetarget mit Features .....	60
Abb. 91: Abbildung von AR-Kamera.....	62
Abb. 92: Abbildung von Imagetarget .....	62
Abb. 93: Inspektor Imagetarget .....	62
Abb. 94: Skalierung Imagetarget.....	63
Abb. 95: ButtonPopup-Skript.....	63
Abb. 96: Canvas über Inspektor an ButtonPopup-Skript zuweisen .....	64
Abb. 97: Abbildung der Build Einstellungen.....	65
Abb. 98: Error: JAVA_HOME .....	65
Abb. 99: Befehl in Eingabeaufforderung.....	65
Abb. 100: Abbildung der Player Einstellungen.....	67
Abb. 101: Abbildung der DPI .....	67
Abb. 102: Abbildung der Stack Trace .....	67
Abb. 103: Abbildung der Bildqualität.....	68
Abb. 104: Unity Profiler.....	69
Abb. 105: Error: Unity rendert nicht .....	70
Abb. 106: Error: Package collab-proxy undefiniert.....	71

Abb. 107: Package collab-proxy neu .....	71
Abb. 108: Package collab-proxy alt .....	71
Abb. 109: OpenGL zurücksetzen .....	73

## Abkürzungen

z.B.	Zum Beispiel
bzw.	beziehungsweise
Vgl.	Verglichen
XML	Extensible Markup Language
DPI	Dots per Inch
URL	Uniform Resource Locator
AR	Augmented Reality
VR	Virtual Reality
JSON	JavaScript Object Notation
JDK	Java Development Kit
APK	Android Package
UI	User Interface
AJAX	Asynchronous JavaScript and XML
API	Application-Programming-Interface
GPU	Graphics processing unit

## Quellen und Literaturverzeichnis

- [1] „Wikipedia Unity,“ [Online]. Available: [https://de.wikipedia.org/wiki/Unity\\_\(Spiel-Engine\)](https://de.wikipedia.org/wiki/Unity_(Spiel-Engine)). [Zugriff am 16. März 2020].
- [2] „Wikipedia Unity Englisch,“ [Online]. Available: [https://en.wikipedia.org/wiki/Unity\\_\(game\\_engine\)](https://en.wikipedia.org/wiki/Unity_(game_engine)). [Zugriff am 01. März 2020].
- [3] „Unity3d-GameObjects,“ [Online]. Available: <https://docs.unity3d.com/Manual/GameObjects.html>. [Zugriff am 01. April 2020].
- [4] „Unity3d-Prefabs,“ [Online]. Available: <https://docs.unity3d.com/Manual/Prefabs.html>. [Zugriff am 01. April 2020].
- [5] „Unity3d-Camera,“ [Online]. Available: <https://docs.unity3d.com/Manual/class-Camera.html>. [Zugriff am 19. März 2020].
- [6] „Unity3d-Sprites,“ [Online]. Available: <https://docs.unity3d.com/Manual/Sprites.html>. [Zugriff am 01. April 2020].
- [7] „Wikipedia C-Sharp,“ [Online]. Available: <https://de.wikipedia.org/wiki/C-Sharp>. [Zugriff am 16. März 2020].
- [8] „Unity3d,“ [Online]. Available: <https://docs.unity3d.com/Manual/UnityCollaborate.html>. [Zugriff am 16. März 2020].
- [9] „Wikipedia Visual Studio,“ [Online]. Available: [https://de.wikipedia.org/wiki/Visual\\_Studio](https://de.wikipedia.org/wiki/Visual_Studio). [Zugriff am 16. März 2020].
- [10] „Wikipedia Visual Studio Englisch,“ [Online]. Available: [https://en.wikipedia.org/wiki/Microsoft\\_Visual\\_Studio](https://en.wikipedia.org/wiki/Microsoft_Visual_Studio). [Zugriff am 01. März 2020].
- [11] „Wikipedia Blender,“ [Online]. Available: [https://de.wikipedia.org/wiki/Blender\\_\(Software\)](https://de.wikipedia.org/wiki/Blender_(Software)). [Zugriff am 15. März 2020].
- [12] „Blender,“ [Online]. Available: <https://www.blender.org/about/>. [Zugriff am 15. März 2020].

- [13] „Blender-Materials,“ [Online]. Available: [https://docs.blender.org/manual/en/2.79/render/blender\\_render/materials/index.html](https://docs.blender.org/manual/en/2.79/render/blender_render/materials/index.html). [Zugriff am 18. März 2020].
- [14] „Blender-Textures,“ [Online]. Available: [https://docs.blender.org/manual/en/2.79/render/blender\\_render/textures/index.html](https://docs.blender.org/manual/en/2.79/render/blender_render/textures/index.html). [Zugriff am 25 März 2020].
- [15] „Blender-Lighting,“ [Online]. Available: [https://docs.blender.org/manual/en/2.79/render/blender\\_render/lighting/index.html](https://docs.blender.org/manual/en/2.79/render/blender_render/lighting/index.html). [Zugriff am 18. März 2020].
- [16] „Wikipedia RabbitMQ,“ [Online]. Available: <https://de.wikipedia.org/wiki/RabbitMQ>. [Zugriff am 15. März 2020].
- [17] „Wikipedia Vuforia,“ [Online]. Available: [https://en.wikipedia.org/wiki/Vuforia\\_Augmented\\_Reality\\_SDK](https://en.wikipedia.org/wiki/Vuforia_Augmented_Reality_SDK). [Zugriff am 16. März 2020].
- [18] „Unity3d-Quaternion,“ [Online]. Available: <https://docs.unity3d.com/ScriptReference/Quaternion.html>. [Zugriff am 20. März 2020].
- [19] „Alloyui,“ [Online]. Available: <https://alloyui.com/examples/color-picker/hsv.html>. [Zugriff am 20. Dezember 2019].
- [20] „Unity-Assetstore,“ [Online]. Available: <https://assetstore.unity.com/packages/tools/gui/runtime-file-browser-113006>. [Zugriff am 08. Jänner 2020].
- [21] „Unity UIBasicLayout,“ [Online]. Available: <https://docs.unity3d.com/Packages/com.unity.ugui@1.0/manual/UIBasicLayout.html>. [Zugriff am 01. März 2020].
- [22] „Unity Canvas-Scaler,“ [Online]. Available: <https://docs.unity3d.com/Packages/com.unity.ugui@1.0/manual/script-CanvasScaler.html>. [Zugriff am 01. März 2020].

- [23] „Wikipedia JSON,“ [Online]. Available: [https://de.wikipedia.org/wiki/JavaScript\\_Object\\_Notation](https://de.wikipedia.org/wiki/JavaScript_Object_Notation). [Zugriff am 09. Jänner 2020].
- [24] „Unity3d,“ [Online]. Available: <https://docs.unity3d.com/ScriptReference/TextAsset.html>. [Zugriff am 05. Februar 2020].
- [25] „Developer-Vuforia,“ [Online]. Available: <https://developer.vuforia.com/downloads/sdk>. [Zugriff am 6. Februar 2020].
- [26] „Unity3d,“ [Online]. Available: <https://docs.unity3d.com/Manual/upm-scoped.html>. [Zugriff am 15. Februar 2020].
- [27] „Developer Vuforia,“ [Online]. Available: <https://developer.vuforia.com/vui/auth/login>. [Zugriff am 23. Februar 2020].
- [28] „Library-Vuforia,“ [Online]. Available: [https://library.vuforia.com/content/vuforia-library/en/reference/unity/classVuforia\\_1\\_1TrackableBehaviour.html](https://library.vuforia.com/content/vuforia-library/en/reference/unity/classVuforia_1_1TrackableBehaviour.html). [Zugriff am 02. März 2020].
- [29] „Answers Unity,“ [Online]. Available: <https://answers.unity.com/questions/1638828/how-to-set-java-home-using-openjdk.html>. [Zugriff am 04. März 2020].
- [30] „Java,“ [Online]. Available: <https://java.com/de/download/faq/develop.xml>. [Zugriff am 15. März 2020].
- [31] „Unity3D-Profiler,“ [Online]. Available: <https://docs.unity3d.com/Manual/Profiler.html>. [Zugriff am 01. April 2020].
- [32] „Unity Forum,“ [Online]. Available: <https://forum.unity.com/threads/unity-games-and-editor-white-screen-problem-might-know-the-real-problem.545441/>. [Zugriff am 25. März 2020].
- [33] „Unity Forum,“ [Online]. Available: <https://forum.unity.com/threads/problem-when-updating-and-changing-the-installation-location.554881/>. [Zugriff am 25. März 2020].
- [34] „Unity3d-manifest,“ [Online]. Available: <https://docs.unity3d.com/Manual/upm-manifestPrj.html>. [Zugriff am 01. April 2020].

- [35] „Unity3d-Packages,“ [Online]. Available: <https://docs.unity3d.com/Manual/Packages.html>. [Zugriff am 01. April 2020].
- [36] „Unity3d-Dependdencies,“ [Online]. Available: <https://docs.unity3d.com/Manual/upm-manifestPrj.html#dependencies>. [Zugriff am 01. April 2020].
- [37] „Unity-Forum,“ [Online]. Available: <https://forum.unity.com/threads/vuforia-augmented-reality-black-screen-when-accessing-android-camera.265335/>. [Zugriff am 01. April 2020].
- [38] „Wikipedia OpenGL,“ [Online]. Available: <https://en.wikipedia.org/wiki/OpenGL>. [Zugriff am 01. April 2020].
- [39] „Reddit\_OpenGL,“ [Online]. Available: [https://www.reddit.com/r/kde/comments/9mlcwr/opengl\\_2\\_vs\\_3/](https://www.reddit.com/r/kde/comments/9mlcwr/opengl_2_vs_3/). [Zugriff am 01. April 2020].
- [40] [Online]. Available: <http://myweb.rz.uni-augsburg.de/~bernt/raster/spektrum/index.shtml?print>. [Zugriff am 15. Jänner 2020].



## **Anhang A – CD-ROM**

Die im Anhang A beteiligte CD-ROM zum Thema „Digitaler Zwilling Ultraschallprüfanlage mit AR“ enthält folgende Inhalte:

- Software
- Dokumentation der Diplomarbeit (dieses Dokument)

## Anhang B – Aufgabenverteilung

Die Ausführung der Aufgaben verteilt sich wie folgt (in Kapitel):

Jakob Deubler:

- 1.1.1
- 2.1
- 2.1.1
- 2.2
- 5 & 6

Aleks Dimitrov:

- 1.1.2
- 1.4 & 1.5
- 2.1.2
- 2.1.3
- 2.1.5
- 2.3
- 2.7
- 7 - 14
- 15.2
- 15.3
- 15.4

Christof Ecker:

- 1.1.3
- 1.2
- 1.3
- 2.1.4
- 2.4
- 2.5
- 2.6
- 3 & 4
- 15.1