Západočeská univerzita v Plzni

Fakulta aplikovaných věd

Katedra kybernetiky

DIPLOMOVÁ PRÁCE

PLZEŇ, 2019 **ONDŘEJ HAVLÍČEK**

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta aplikovaných věd
Akademický rok: **2018/2019**

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Ondřej HAVLÍČEK**
Osobní číslo:	**A16N0143P**
Studijní program:	**N3918 Aplikované vědy a informatika**
Studijní obor:	**Kybernetika a řídicí technika**
Název tématu:	**Integrace rozšířené reality do postupu virtuálního uvádění do provozu**
Zadávající katedra:	**Katedra kybernetiky**

Z á s a d y   p r o   v y p r a c o v á n í :

Navrhněte řešení pro integraci rozšířené reality v procesu virtuálního uvádění do provozu. Zaměřte se na propojení softwarových programů NX MCD, herního enginu Unity a brýle pro rozšířenou realitu Microsoft HoloLens. Výsledné řešení by mělo být schopno vizualizovat model z NX MCD pomocí Microsoft HoloLens. Vypracujte pracovní postup pro využití navrhovaného řešení.
1. Prostudujte pracovní proces virtuálního uvádění do provozu.
2. Navrhněte možná řešení převedení mechatronických modelů z NX MCD do rozšířené reality za pomocí Unity a HoloLens.
3. Po konzultaci s vedoucím práce jedno řešení implementujte.
4. Zakomponujte vaše řešení do pracovního procesu virtuálního uvádění do provozu.
5. Pro implementované řešení vypracujte pracovní postup pro jeho snadné využití.
Práci konzultuje se zástupci ze společnosti Siemens s.r.o: Dmitry Kochubey (dmitry.kochubey@siemens.com), Harald Funk (harald.funk@siemens.com), Tobias Koedel (tobias.koedel@siemens.com)

Rozsah grafických prací:	**dle potřeby**
Rozsah kvalifikační práce:	**40-50 stránek A4**
Forma zpracování diplomové práce:	**tištěná**

Seznam odborné literatury:

**1. Unity (2018). Unity for VR and AR [OB/OL]. [2018-10-07].
https://unity3d.com/unity/features/multiplatform/vr-ar
2. Siemens AG (November 1, 2017). SIMATIC S7 PLCSIM Advanced: Co Simulation via API, Industry Online Support [DB/OL]. [2018-10-07].
https://support.industry.siemens.com/cs/ww/de/view/109739660
3. Mono (2018). The Mono Runtime [OB/OL]. [2018-10-07].
https://www.mono-project.com/docs/advanced/runtime/
4. Siemens AG (August 3, 2018). Digitalization in industry - Twins with potential [DB/OL]. [2018-10-07]. https://www.siemens.com/customer-magazine/en/home/industry/digitalization-in-machine-building/the-digital-twin.html**

Vedoucí diplomové práce:	**Ing. Ondřej Severa**
	Výzkumný program 1

Datum zadání diplomové práce:	**1. října 2018**
Termín odevzdání diplomové práce:	**25. května 2019**

Radová

Doc. Dr. Ing. Vlasta Radová
děkanka

Západočeská univerzita v Plzni * Fakulta aplikovaných věd *

L.S.

[illegible]

Prof. Ing. Josef Psutka, CSc.
vedoucí katedry

V Plzni dne 1. října 2018

Poděkování

Tímto bych chtěl poděkovat vedoucímu mé diplomové práce, panu Ing. Ondřeji Severovi, za jeho odborné vedení, za podnětné návrhy k práci a za čas, který mi byl ochoten věnovat. Dále bych také rád poděkoval panu Ing. BSc Dmitry Kochubey, za jeho vedení, rady a pomoc při řešení odborných problémů během diplomové práce a také panu Dipl.-Ing. Johannes Heinrich Hofmann, který mi umožnil uskutečnění této diplomové práce ve spolupráci se společností Siemens. Na závěr bych chtěl poděkovat mé rodině za to, že mi poskytla výborné zázemí pro studium na vysoké škole a psaní této diplomové práce.

Abstrakt

Tato práce se zabývá integrací rozšířené reality do procesu virtuálního uvádění do provozu. Práce popisuje jednotlivé kroky obvyklého uvádění do provozu a posléze popisuje kroky modernějšího procesu virtuálního uvádění do provozu, který se snaží využívat nově vznikajících technologií. Na několika skutečných příkladech demonstruje využitelnost virtuální a rozšířené reality v průmyslu a následně navrhuje, jak využít rozšířenou realitu v podobě Microsoft HoloLens brýlí v procesu virtuálního uvádění do provozu. Navržené řešení zobrazí pracovní zóny a 3D model v HoloLens aplikaci, která po zobrazení umožní i úpravu těchto zón a synchronizuje je s ostatními programy. Výsledné řešení se skládá ze serveru, který běží na PC, a HoloLens aplikace, která komunikuje s tímto serverem přes Wi-Fi pomocí TCP soketů. Aplikace využívá funkcí programu NX MCD, kde se vytváří a exportují modely, a programu TIA Portal, kde se definují pracovní zóny, se kterými se pak následně pracuje v řídícím systému.

Klíčová slova: HMI, HoloLens, Rozšířená realita, Virtuální uvádění do provozu.

Abstract

This thesis investigates the integration of augmented reality into the digital workflow of virtual commissioning. The thesis describes the stages of regular workflow of commissioning and then it describes the improved process of the virtual workflow of commissioning that uses newly emerging technologies. It demonstrates the applicability of virtual and augmented reality applications on several real-life use cases and then it proposes a solution to utilize the augmented reality via Microsoft HoloLens smartglasses. The proposed solution visualizes workspace zones and a 3D model using the HoloLens application that enables to adjust these workspace zones and synchronize them with other programs. The final solutions then consist of a server application running on a PC and HoloLens application that communicates with the server over Wi-Fi using TCP sockets. The application uses functions from the NX MCD program where 3D models are created and exported, and from TIA Portal where the workspace zones are defined and then used in a control system.

Keywords: HMI, HoloLens, Augmented Reality, Virtual commissioning.

Table of Contents

1 Introduction

Today, the rapidly changing marketplace demands that a manufacturing system should be able to swiftly adapt to a wide range of circumstances. Therefore, a modern manufacturing system should have sufficient responsiveness to adapt their behavior and at the same time be able to fulfil the constraints given by the cost, shortening product lifecycle, and strategies for rapid time-to-market. For these reasons, the timeframe for creating and commissioning a manufacturing systems tightens whereas the demand on planning accuracy and planning quality grows [1], [2].

The workflow of commissioning a manufacturing system can be in general divided into several stages. Concept Design stage, Mechanical Design stage, Electric Design stage, Automation programming stage and Commissioning stage [2]. The conventional commissioning workflow is usually done in a linear sequence where each stage is done after the previous one is completed. For that reason, an integrated test of the manufacturing systems cannot be done before the real commissioning. Consequently, many design mistakes are discovered during the commissioning which leads to time and money consuming corrective measures. Therefore, a new workflow of commissioning, called virtual commissioning, has been researched for more than 10 years [2]. The virtual commissioning creates a new stage in the system development workflow for more validation capabilities. Both of these workflows are described in detail in section 1.1. The section 1.2 provides a description how these workflows are implemented using a software solution provided by Siemens company.

One of the new technologies that could improve the overall workflow of commission is virtual and augmented reality. The thesis describes several real-life use cases of virtual and augmented reality and based on these examples, it proposes a solution that integrates augmented reality into the virtual commissioning workflow. The goal of this thesis is to seamlessly integrate the augmented reality into digital workflow of virtual commissioning in order to improve the overall product life cycle management.

The thesis is structured as follows. The first chapter contains the thesis introduction and describes the workflow of virtual commissioning and how this workflow is implemented in Siemens company products. Furthermore, it describes different real-life use cases that are implemented in virtual and augmented reality. The next chapter 2 defines and analyses the problem that this thesis will address. The following chapter 3 gives a general description of the technologies that were used in this thesis. The chapter 4 describes the kinematics and the problems with describing position and rotation of an object in a general 3D space. The following chapter 5 then describes the proposed solution based on the use case specification that was provided in the chapter 2. The chapter 6 then demonstrates the application features and capabilities.

The thesis ends with the evaluation of the improved workflow of commissioning with the developed application and then discusses possible future work and improvements.

1.1 Workflow of Commissioning

The workflow of commissioning is a process of developing and commissioning a plant or a line, based on customer's requirements and needs. The main objective is to automate a production line to improve productivity, quality and reduce the costs of maintenance.

Before commissioning a plant there are several design stages that must be done. This includes Concept Design stage, Mechanical Design stage, Electric Design stage and Automation programming stage. After these four stages are complete, it is then possible to start commissioning a plant. During these design stages the engineers try to propose, to the best of their knowledge, a solution that will fulfill the defined requirements. The workflow of all these stages as well as the commissioning stage is sequential which corresponds to a Waterfall model [3]. A Waterfall model is a type of Software Development Life Cycle (SDLC) process [3]. This means that in theory, each stage should be completed before the following stage begins as is visualized in the following Figure 1.1.

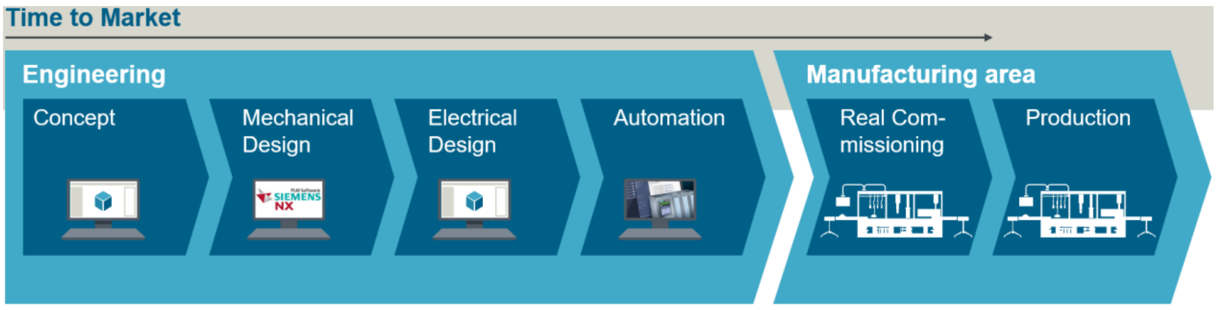


Figure 1.1 Timeline of the workflow of commissioning

The Waterfall model is known to not be very iterative which can lead to unexpected mistakes in design that are very expensive at later stages of the workflow.

According to Peter Hoffmann et al. who reference to a study of the German Association of machine tool builders, the commissioning time uses up to 25 % of the available time for plant engineering and construction; and up to 15 % is consumed by correcting mistakes in the control software [2].

Therefore, a new workflow is being proposed that is called workflow of virtual commissioning. This new workflow adds a stage before the real commissioning and tries to do the Mechanical Design, the Electrical design and the Automation design in parallel as shown in Figure 1.2.

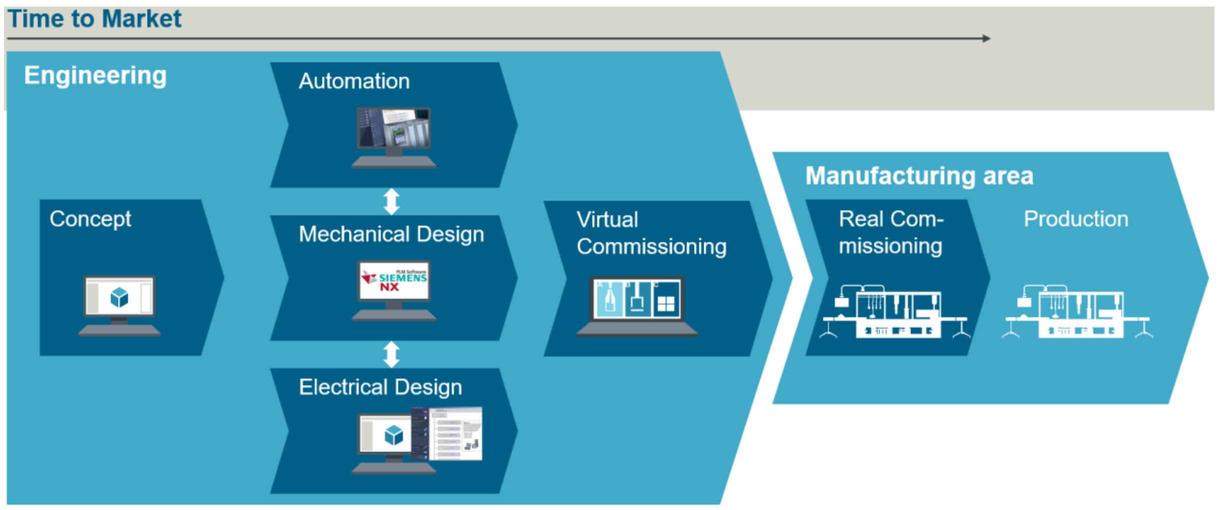


Figure 1.2 Proposed timeline of the virtual workflow of commissioning

The workflow with parallel processes corresponds more to the Agile development model [4] which is more common in software development because of its iterative behavior. However, the main advantage virtual commissioning workflow is that the engineers will test functions of the plant in virtual reality. Because of this feature, many mistakes during design can be discovered and rapidly redesigned with almost no cost compared to real commissioning. This advantage can be visualized in “the cost of change” curve in Figure 1.3. When using the Waterfall model, the cost of changing the design requirement rises exponentially, therefore in the early stages of development (i.e. in concept design or electric design) any changes are quite cheap because the automation engineers have not created any software based on the concept design. Changes in Agile development are obviously more expensive at this point. This will change in the long run, especially when a major mistake in design during the Automation stage is detected. If the virtual commissioning stage is successful and the systems were tested in virtual reality, it is reasonable to expect that less changes will occur in later stages of the SDLC [4].

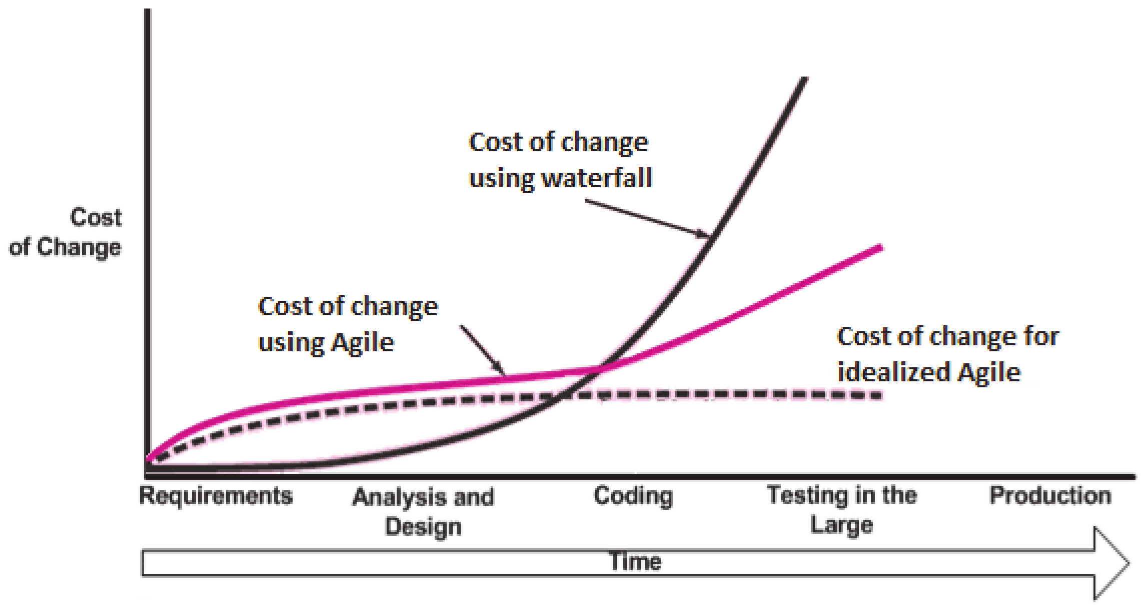
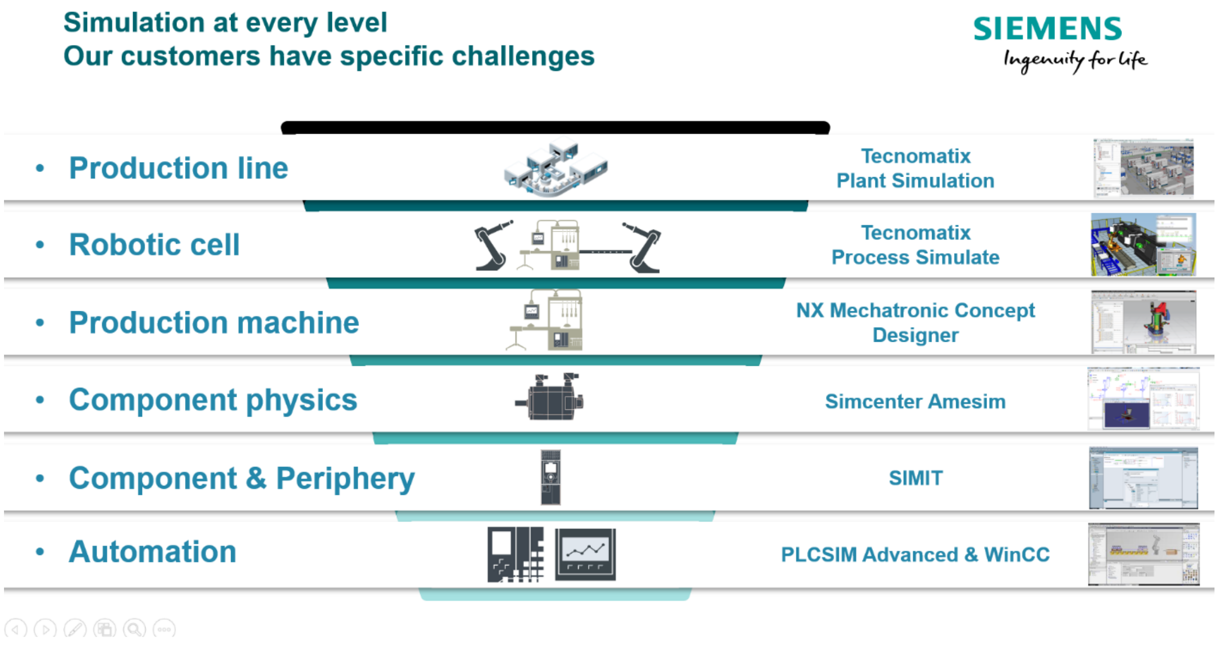


Figure 1.3 Cost of change over time in different SDLC processes [4]

In addition, thanks to this iterative behavior and early detection of mistakes, the time needed to commission a plant shortens and the quality of the product rises.

1.2 Siemens's Virtual Workflow of Commissioning

Siemens provides a wide range of software solutions to cover process of the virtual workflow commissioning based on the customer needs. The Figure 1.4 shows several customer use cases and describes which Siemens software is used for each particular case.


Simulation at every level
Our customers have specific challenges
SIEMENS
Ingenuity for life

Production line	Tecnomatix Plant Simulation
Robotic cell	Tecnomatix Process Simulate
Production machine	NX Mechatronic Concept Designer
Component physics	Simcenter Amesim
Component & Periphery	SIMIT
Automation	PLCSIM Advanced & WinCC

Figure 1.4 Siemens solutions for simulation based on the customer needs

An example how the workflow can be implemented using the Siemens software is described in the Figure 1.5.

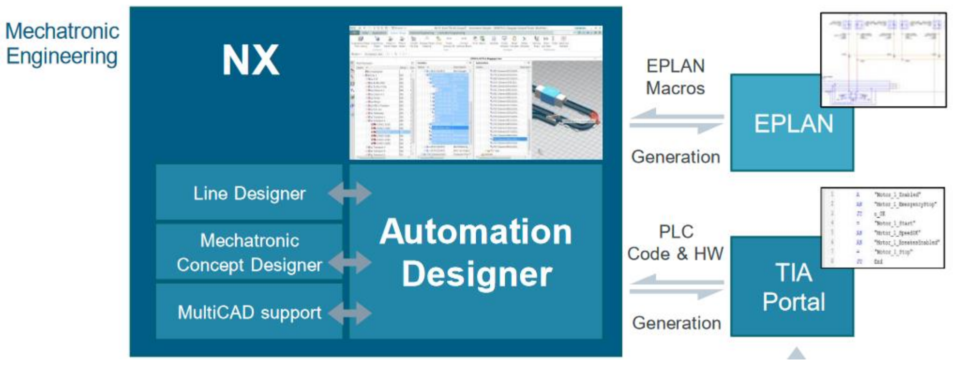


Figure 1.5 Interaction between different stages in the workflow of commissioning in Siemens solutions [5]

When examining the Figure 1.2 and Figure 1.5 the Mechanical Design stage is controlled by NX which is a specialized Siemens CAD, CAE, CAM software that allows to create a mechanical part or its concept using CAD; verify and check the mechanical properties via CAE; produce a part and check its quality using CAM. The electric design is usually done by a third-party solution like EPLAN [6] which is then integrated into TIA Portal using an open standard called Automation Markup Language (AutomationML). The Automation stage is developed using the TIA Portal where the control system is programmed.

At the heart of the interaction between these stages is a Siemens's Automation designer which provides central design application for electrics and automation software in the Digital Enterprise [5].

The stage of virtual commissioning is done by integrating the Mechanical Design in NX and the TIA Portal control system by adding a kinematic model to NX's mechanical design. This kinematic model can then be controlled by a real PLC or PLCSIM Advanced which is Siemens program for simulating PLC behavior. The mechanical model in NX can now be controlled from outside the NX and the user is able to evaluate the 3D model as well as the controlling program using the Software-In-the-Loop (SIL) simulation.

1.3 Virtual Reality Use Cases

The virtual reality displays come in almost any configuration imaginable and virtual reality facilities may use one or more of the following technologies. A single large projection screen commonly called Powerwall; a multiple connected projected screen to give the user the feeling of being inside virtual reality commonly called CAVE (Cave Automatic Virtual Environment) [7]; stereo-capable monitors with desktop tracking and head-mounted display (HMD).

Several manufacturers of virtual reality headsets exist such as HTC VIVE by the Valve Corporation [8], Oculus Rift owned by Facebook [9], and the Samsung Gear VR [10]. They all obscure the user's vision and create a virtual reality experience. This technology can be used for many use cases, such as developing and designing a robot or a machine. The user can then get a better idea of the machine size and even perform some kinematic simulations to test and debug the control software. This is all done in virtual reality, so any mistakes or design changes are safe and done almost free of charge compared to mistakes that happen during the commissioning of a real machine.

A survey of industry use of virtual reality in product design and manufacturing by Berg, L.P. & Vance, J.M. [7] across several industries revealed that the most common scenario use case is evaluating the human ability to see in a particular setting or posture. This is well suited for VR applications because the designer can get a true sense of what the human can see.

Another common use case was to study ergonomics and reachability. The designer asked questions like: "How's someone going to posture themselves to do this technique?". At Ford Motor Co., ergonomic engineers use VR to find the right design criteria related to the maximum allowable assembly force to install various hoses. Using the HMD, physical props and force sensors, the engineers estimate the forces required to mount hoses given certain human posture.

1.4 Augmented Reality Use Cases

According to Akshay Kore [11] Augmented reality (AR) devices can be divided into four different groups as visualized in the following Figure 1.6.

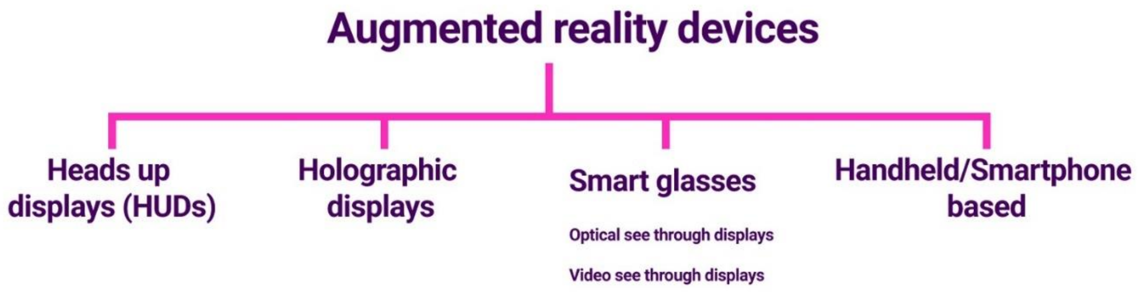

Augmented reality devices
Heads up displays (HUDs)
Holographic displays
Smart glasses
Optical see through displays
Video see through displays
Handheld/Smartphone based

Figure 1.6 Augmented reality devices division [11]

The author of this thesis would argue that video see-through displays are not augmented reality devices because they do not extend the actual reality but rather record the reality and then render digital content to a screen to create the impression of a hologram. Otherwise the author agrees with this division.

Heads up displays (HUD) can be a transparent display that renders some information on it. For example, imagine an army fighter plane screen that displays the current rotation of an airplane. Holographic displays are the “Star Wars” holograms we would expect. They do not require users to wear any gadgets. Smart glasses with optical see through displays are a popular augmented reality device. As the name suggests they augment or extend reality with new content. They render digital content to transparent glasses to create the impression of holograms. The last device is handheld, or smartphone based. The rise of new AR libraries like ARKit (iOS), ARCore (Android) or Mixed Reality ToolKit – MRTK (Windows) make creating augmented reality experiences even easier which might lead to new interesting use cases.

One of the use cases where augmented reality would help could be civil engineering as described in Siddhant Agarwal’s article [12]. He refers to the work of Rankohi and Waugh [13] who showed that field workers and project managers have a high interest in using non-immersive and desktop standalone AR technologies during the project construction phase, mainly to monitor progress and detect defective work. He describes the problem following way: “*Currently, the engineers in the field rely on scaled drawings in all projects that are used for all practical purposes. And this is where human error tends to creep in the execution*” [13] . He discusses the possible benefits of implementing AR technology in civil engineering such as reducing human error thanks to better control of different processes and better marketing thanks to project visualization.

Today, the most well-known real-life AR use cases are mobile games like Pokémon-Go which is the most downloaded iOS app of 2016 [14]. We might also expect a boom of applications for smart glasses thanks to the Microsoft HoloLens which seems to be gaining popularity in the developer community, but these applications are probably going to be more business-oriented [15].

One of the interesting real-life applications is IKEA’s smartphone app for furniture shopping. It is possible to list the catalog and place the furniture in your room to get the idea of how it might look using a smartphone. If the customers use this application, they do not necessarily have to measure the size of the room and the furniture and then calculate whether the furniture fits there [16].

2 Problem Definition and Analysis

The convetional approach to commissioning described in the previous chapter does not take full advantage of modern technologies regarding the virtual and augmented reality. The workflow of virtual commissioning could take inspiration from the real-life use cases described in previous chapter. These use cases utilize the virtual and augmented reality to improve the SLDC process as well as the overall product life cycle. Based on the studies mentioned above, there is a wide range of use cases for mixed reality applications, therefore it is reasonable to utilize the new technologies in the Siemens workflow of virtual commissioning.

2.1 Problem Definition

The current Siemens virtual commissioning workflow does not offer an easy way to integrate any of its software solutions for virtual commissioning with augmented reality. For example, if the user wants to export a model to augmented reality, he has to do all the necessary tasks manually as shown in the Figure 2.1.

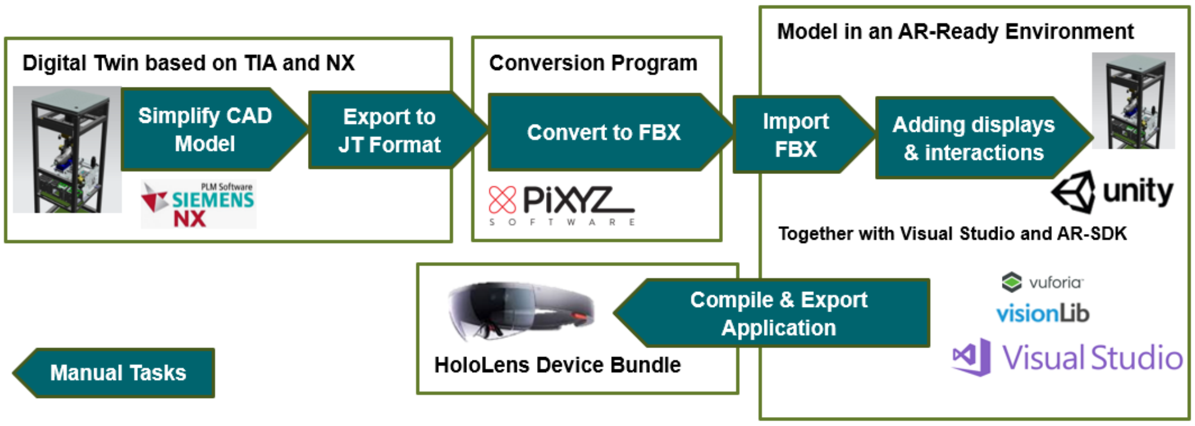


Figure 2.1 Task required to Export a 3D model from

Currently there is no simple way to utilize an augmented reality glasses such as Microsoft HoloLens in the system development workflow which might be useful for the user. Furthermore, a third-party solution must be used in process of simplifying a 3D model in order to be usable in augmented reality applications.

2.2 Problem Analysis

Based on the problem definition an application that could automatically export 3D models from NX and import them to HoloLens glasses is needed. The application should also be able to integrate other Siemens software solutions such as TIA Portal in order to take the full advantage of the augmented reality potential.

This thesis proposes a solution that would export 3D models from NX and integrate work space zones into the workflow of virtual commissioning. The definition and functionality of work space zones are described in detail in chapter 3.6 but in general they define a work space where the machine can or cannot move. These zones are defined in TIA Portal and could provide a general description of the real space so that the commissioning engineer can get the general idea of the space distribution. It would also be convenient if he could adjust the zones in real time and upload those changes back to the TIA Portal.

These two use cases would also be beneficial for the automation engineer who programs the PLC during the development stage. This would be helpful because it allows to view the defined zones and the developed 3D model in NX at the same time. The defined use case diagram is visualized in the following figure.

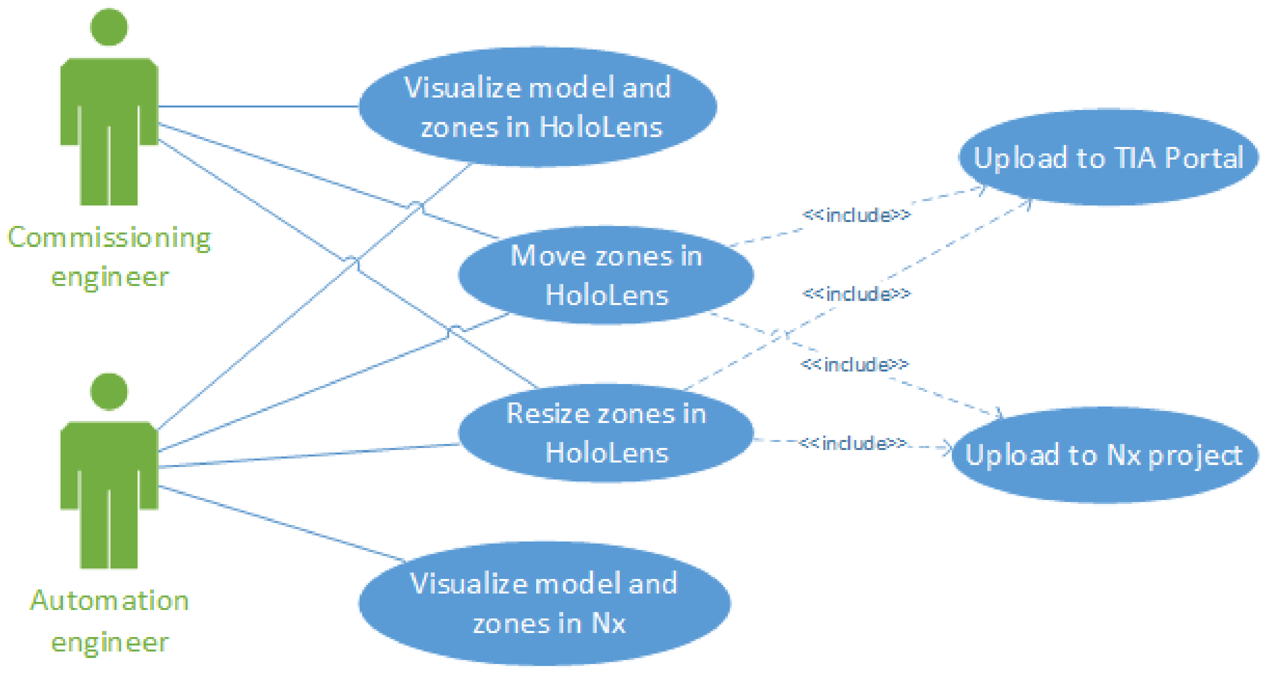


Figure 2.2 Use Case diagram

The general idea is that the automation engineer has the possibility to export the zones from the TIA Portal to NX and HoloLens and visualize them in these software solutions. The automation engineer needs to have an option to modify the zones in HoloLens application and synchronize the data back to the TIA Portal and NX.

The commissioning engineer on the other hand, does not need to visualize the zones in NX but when he does change something it is important to have the zones synchronized between the TIA Portal, NX and HoloLens application.

2.2.1 Use Case Specification

Use Case Name:	Visualize model and zones in HoloLens
Actor(s):	Commissioning engineer, Automation engineer
Description:	Visualize 3D model and work space zones in the HoloLens application
Pre-Condition:	The Server is ready
Basic Path:	1. The user starts application in HoloLens 2. The user finds QR code representing reference point with camera 3. The app loads the default 3D model 4. The user opens console 5. The user loads model using the console

Table 2-1 Use case specification - Visualize model and zones in HoloLens

Use Case Name:	Move zones in HoloLens
Actor(s):	Commissioning engineer, Automation engineer
Description:	Move the work space zones in HoloLens application
Pre-Condition:	The "Visualize model and zones in HoloLens" use case basic path was done
Basic Path:	1. Change to a move mode using menu bar 2. Change position of the zone using hand 3. Change to a default mode using menu bar

Table 2-2 Use case specification - Move the work space zones in HoloLens application

Use Case Name:	Resize zones in HoloLens
Actor(s):	Commissioning engineer, Automation engineer
Description:	Resize the work space zones in HoloLens application
Pre-Condition:	The "Visualize model and zones in HoloLens" use case path was done
Basic Path:	1. Change to a resize mode using a menu bar 2. Change size of the zone using hand 3. Change to a default mode using menu bar

Table 2-3 Use case specification - Resize zones in HoloLens

Use Case Name:	Upload to TIA Portal
Actor(s):	Commissioning engineer, Automation engineer
Description:	Resize the work space zones in HoloLens application
Pre-Condition:	The “Resize zones in HoloLens” use case or “Move zones in HoloLens” use case was done
Basic Path:	1. User opens console 2. Uploads new data to server using console

Table 2-4 Use case specification - Upload to TIA Portal

Use Case Name:	Upload to NX project
Actor(s):	Commissioning engineer, Automation engineer
Description:	Resize the work space zones in HoloLens application
Pre-Condition:	The “Resize zones in HoloLens” use case or “Move zones in HoloLens” use case was done
Basic Path:	1. User opens console 2. Uploads new data to server using console

Table 2-5 Use case specification - Upload to NX project

Use Case Name:	Visualize model and zones in NX
Actor(s):	Automation engineer
Description:	Export work space zones data from TIA Portal to NX 3D model
Pre-Condition:	• TIA Portal project with Kinematic Technological Object exists • NX project exists
Basic Path:	1. The user defines input arguments for application in XML file 2. The user starts application 3. The application exports the zones to NX

Table 2-6 Use case specification - Visualize model and zones in NX

3 Technologies Used

This chapter contains a description of all used technologies that are necessary for a full understanding of the proposed solution. A list of the software solutions and frameworks used is listed in following Table 3-1.

Software/Hardware	Version	License
Unity	2017.4.23f1	Community license
Vuforia Engine	7.0.57	Developer license
HoloLens	HoloLens 1st gen	
HoloToolkit	2017.4.3.0 - Refresh	MIT license
Visual Studio 2017	15.9.11	Community license
TIA Portal	V15.1	
NX MCD	12.0	

Table 3-1 List of used hardware and software

3.1 Unity

Unity is a cross-platform real-time game engine for developing various games and applications. It is developed by Unity Technologies which was founded in 2004 [17]. Unity is the most widely used VR and AR development platform and is used in the creation of over 91% of HoloLens experiences [18].

Unity can be used to develop 2D and 3D games as well as simulations using unity physics for wide range of platforms. The engine supports more than 25 platforms across mobile, desktop, console, TV, VR, AR and the Web including Microsoft HoloLens. The Figure 3.1 gives an overview of the supported platforms.

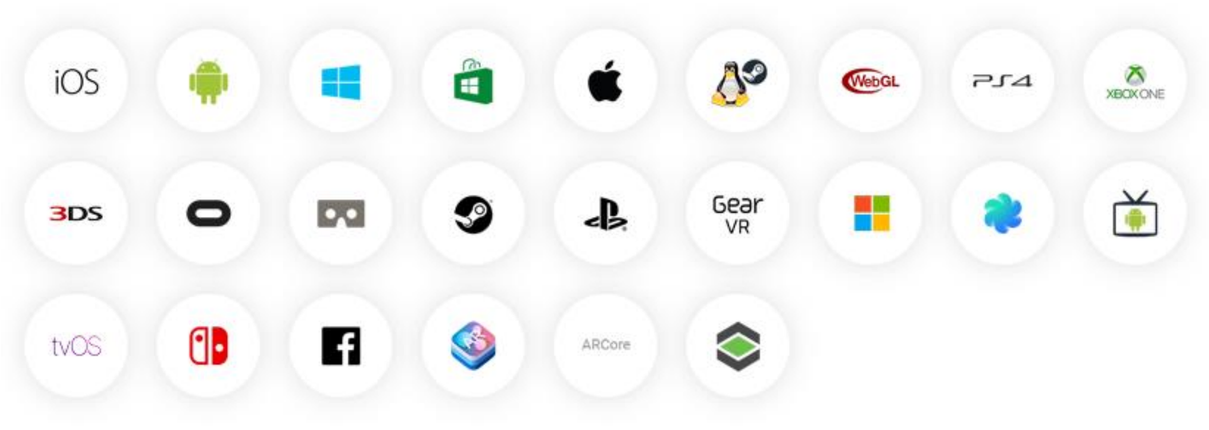


Figure 3.1 Overview of Unity's supported platforms [18]

3.1.1 Unity editor

One of the main advantages of using the Unity game engine is that it comes with an editor which is an essential part when working with the Unity engine. The editor integrates all technologies used and provides a convenient graphical user interface for the developer.

Unity uses C# as a scripting language. When developing a game in Unity the general idea is that you divide the game in scenes (sometimes referred to as levels) and then add game objects to these scenes. Game objects are essentially a capsule for some feature, service or behavior. It is up to the developer to decide how to implement his game.

Each game object has components which are one of the most important things in Unity because they make up a huge part of the game. By default, each game object must have a `Transform` component which defines an object's position, rotation and scale in a scene. The component can be anything that behaves or acts in a scene. That is why, when writing C# script, you must inherit from `MonoBehaviour` class in order to be able to attach the component to a game object. A special game object that can be stored in an asset folder is called Prefab. This is like a template game object with defined components and parameters that you can instantiate a use within Unity editor or even during runtime.

Each `MonoBehaviour` class has methods that are based on Unity's defined execution order which is available in the manual [19]. More on how to develop in unity can be found on Unity's official pages. There are also great resources on Unity learn page, the Unity Manual, and the Unity scripting reference page [20], [19].

3.2 Vuforia Engine

Vuforia Engine is a platform for AR development with support for leading phones, tablets, and eyewear. It is developed by Parametric Technology Corporation (PTC) which focuses on developing software to improve product lifecycle management (PLM) including the integration of augmented reality solutions into industrial use cases.

The Vuforia Engine uses computer vision technology to recognize and track planar images (Image Targets), ground planes in user's environment, 3D objects in real time. The engine supports the Unity Engine as well as three major native platforms: iOS, Android and UWP (Universal Windows Platform). The Vuforia Engine detects the capabilities of underlying devices and fuses them with Vuforia Engine features. The process of fusing different data is called Vuforia Fusion. Figure 3.2 gives an idea of the different parts of the Vuforia Engine.

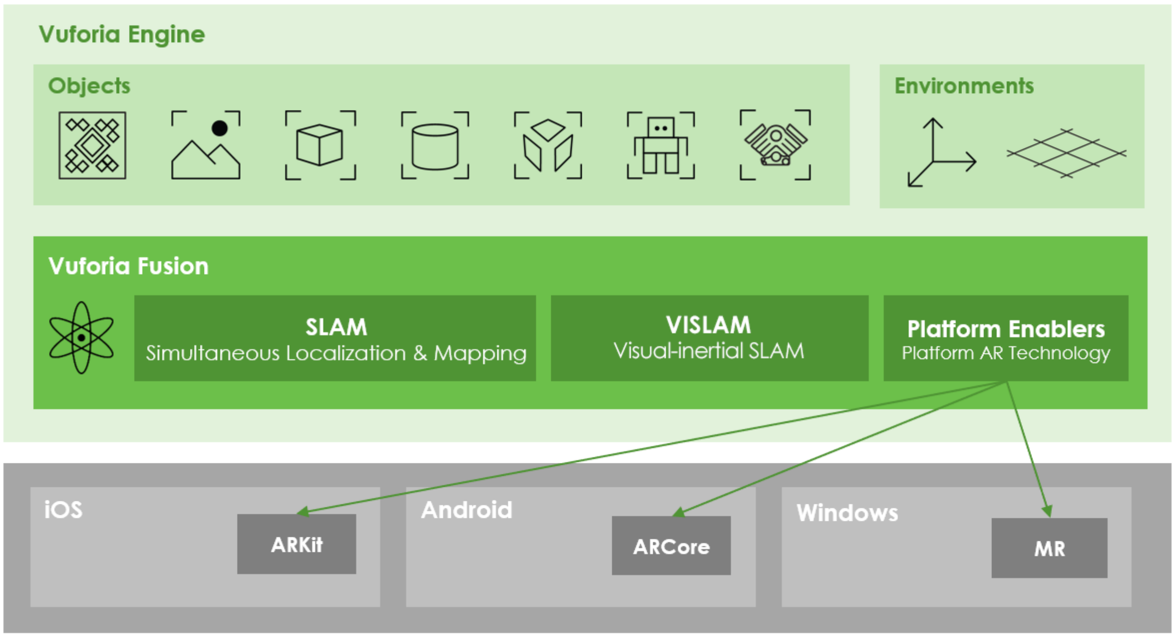


Figure 3.2 Vuforia Fusion components [21]

The Vuforia Fusion capability is a procedure that solves the problem of fragmentation in AR-enabling technologies on different platforms. This means that the Vuforia Fusion tries to leverage the richest set of software/hardware enablers as described in Figure 3.3.

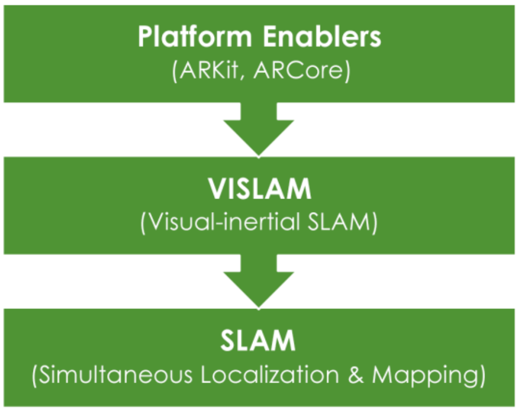


Figure 3.3 Process of detecting available tools in Vuforia Fusion [21]

First Vuforia Fusion tries to find a specialized SDK for AR. In practice this means to find either ARKit (iOS), ARCore (Android) or MR – Mixed Reality (Windows). If not found, the Vuforia Fusion tries to utilize Visual Inertial Simultaneous Localization & Mapping (VISLAM). The SLAM is a general problem of constructing or updating a map of an unknown environment while simultaneously keeping track of an agent's location within it [22]. The VISLAM is an algorithm implemented by Vuforia combining the benefits of Visual-Inertial Odometry (VIO) and Simultaneous Localization and Mapping (SLAM) [21].

The term Visual Odometry (VO) represents a problem of estimating the position and orientation of a camera-carrying platform by analyzing images taken from consecutive poses. The VIO has the same problem but it includes data gathered from inertial sensors to the estimation algorithm [23].

3.3 HoloLens (1st gen)

The first-generation HoloLens are mixed reality glasses that create the impression of holograms. From the developer's point of view, it is a standalone computer with ordinary components like batteries that last for 2-3 hours of active use, 64GB Flash memory, 2 GB RAM, 32-bit Intel CPU (1 GHz), micro USB 2.0, built-in speakers, Bluetooth and Wi-fi. The glasses have a default HD resolution 720p (1268x720) but it is possible to lower the resolution to 360p (634x360). The HoloLens also has a custom-made Holographic Processing Unit (HPU) which is optimized for heavy computation that is needed for rendering holographs [24], [25].

Figure 3.4 Microsoft HoloLens smartglasses

3.3.1 Mixed Reality

Microsoft defines mixed reality as the result of blending the physical world with the digital world. The mixed reality concept can be described as a spectrum of two realities, the physical reality on one side and the digital reality on the other. The following figure describes the blending between these two realities.

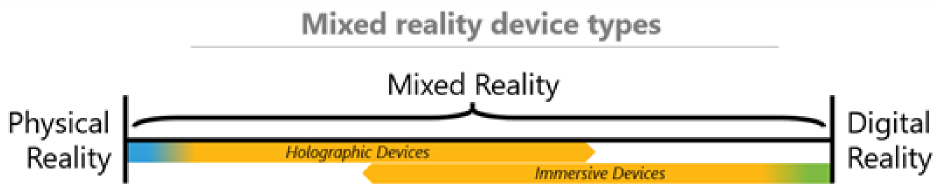


Figure 3.5 Mixed reality device types [25]

On the right side the user experiences a completely digital environment and is oblivious to what occurs in the physical environment around him. On the left side the user is made to believe that he never left the real environment. The middle area between physical reality and digital reality represents a smooth blending between the real world and the digital world. Microsoft compares this middle point as the experiences that the characters in the movie Jumanji (1995) had.

When Microsoft talks about development for HoloLens it uses the terms “holographic” and “holograms”, but this could be misleading. The HoloLens eyewear does not fulfill the definition of hologram because real holograms do not require you to wear special goggles to see them. Scott Stein from cnet.com has a perfect way of describing this problem: “They (HoloLens) do not meet the "Help me, Obi-wan Kenobi" test, nor the dictionary definition of hologram” [26]. The dictionary definition of hologram according to merriam-webster.com is “a three-dimensional image reproduced from a pattern of interference produced by a split coherent beam of radiation (such as a laser)” [27]. It is therefore more accurate to refer to HoloLens rendered objects as digital content or augmented reality content but not holograms.

3.3.2 HoloToolkit

As described in the official documentation, the HoloToolkit is a collection of scripts and components intended to accelerate the development of applications targeting Microsoft HoloLens and Windows Mixed Reality headsets. The HoloToolkit is available on GitHub under the MIT license but it has been deprecated in favor of the Mixed Reality Toolkit (MRTK) since December 5, 2018 [28]. The development of this thesis’s HoloLens application began in December 2018 and since there has been available only pre-releases of MRTK it was decided to use HoloToolkit instead of MRTK. As of now (April 2019), there is still no stable release available.

3.4 Sockets

The communication over a network between the HoloLens application and the server application was achieved by using an API for Inter Process Communication called Berkeley Sockets (often referred to as BSD Sockets). The BSD Sockets were popularized by Berkeley Software Distribution (BSD) UNIX and is a common two-way communication between processes.

Thanks to the popularity of these sockets, there exist many libraries implementing this API. The implementation the BSD sockets are also included in the .NET Framework and the Universal Windows Platform (UWP) [29], [30]. Both of these implementations were used in this thesis because the author of this thesis had more experiences with .NET Framework and therefore, he decided to implement the server application in .NET Framework. Unfortunately, the HoloLens application does not support the .NET library and therefore, he had to use the UWP implementation of BSD sockets. Both of these libraries implement Transmission Control Protocol (TCP) and (User Datagram Protocol) UDP sockets. A TCP socket was used for communication between the applications over the Wi-Fi network.

3.5 Pipes

A pipe is a section of shared memory that processes use for communication. As the name suggests it has two ends. One process writes data and the other process reads data and therefore, it is a simplex communication (one-way communication). In order to have duplex communication (two-way communication) it is needed to create two pipes and give the other process handles for both pipes.

The .NET Framework has two implementations of pipes at its disposal. The first is called the Named pipe and the second is the Anonymous pipe. The Named pipe provides an interface for transferring data between processes on the same computer or processes that communicate over a network. An Anonymous pipe provides an interface for transferring data as well, but it typically transfers data between a parent process and a child process. Anonymous pipes are always local, and they cannot be used for communication over a network. Compared to named pipes, the Anonymous pipes require less overhead. An Anonymous pipe exists until all pipe handles have been closed. All pipe handles are also closed when the process terminates. [31].

In this thesis a .NET Framework implementation of Anonymous pipe is used.

3.6 TIA Portal

Totally Integrated Automation Portal (TIA Portal) is part of the SIMATIC product family. It is an editor that integrates several other SIMATIC software products including SIMATIC STEP 7 for programming PLCs (Programmable Logic Controllers), and WinCC for designing HMIs (Human Machine Interfaces). It provides a convenient way for developing automation solutions.

One of the features of PLC programing in the TIA Portal is programming kinematics technology objects. These are special objects for calculating the motion setpoints for the tool center point (TCP) with respect to the robot's dynamics settings. The kinematics technology objects can calculate motion setpoints using the forward and inverse kinematic transformation based on the robot composition. The kinematic model is defined within the kinematic technology object along with several coordinate systems and zones. The list of coordinate systems defined in kinematic technology object is listed in Table 3-2**.**

Name	Position
World coordinate system (WCS)	Fixed coordinate system of the environment or workspace of the kinematics.
Kinematics coordinate system (KCS)	Defined relative to WCS
Flange coordinate system (FCS)	Defined relative to KCS
Tool coordinate system (TCS)	Defined relative to FCS
Object coordinate system (OCS)	Defined relative to WCS

Table 3-2 Types of Coordinate systems defined in Technology Objects

The following Figure 3.6 was extracted from the official TIA Portal documentation of Kinematics Functions [32]. It demonstrates the relative position of coordinate systems.

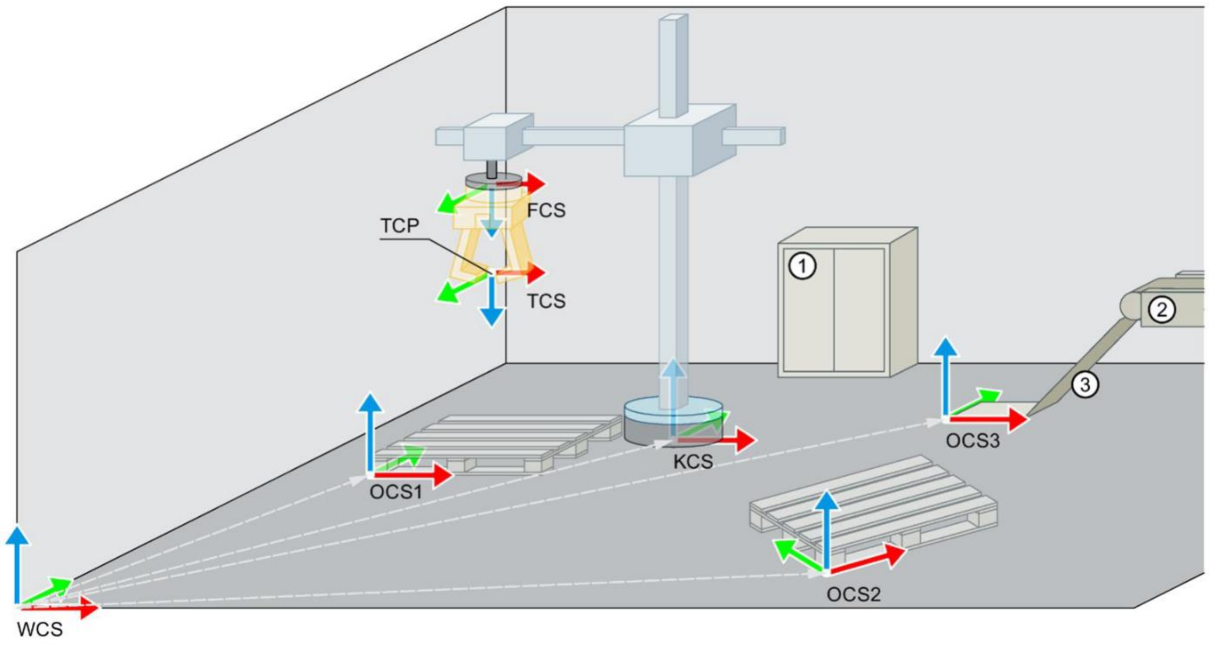


Figure 3.6 Example of coordinate system placement in Technology Object

Another feature of kinematic technology object is zone monitoring. The purpose of zone monitoring is to avoid collision with mechanical installations and triggering process related actions. In order to use this feature, you must define workspace zones, which describes the environment, and kinematics zones, which describe the shape of the machine. The zone monitoring checks the kinematics zones for penetration with the workspace zones.

The workspace zones further divide into 3 types:

1. Work zones

   The work zone is a space where the kinematic zones can move freely, and no signal will be issued. When the work zone is not defined, the whole environment is a work zone.
2. Signal zones

   When a kinematic zone enters a signal zone, a signal is triggered to the user program
3. Blocked zones

   The blocked zones will trigger an alarm and stop the machine.

The following figure demonstrates the use of kinematics and workspace zones.

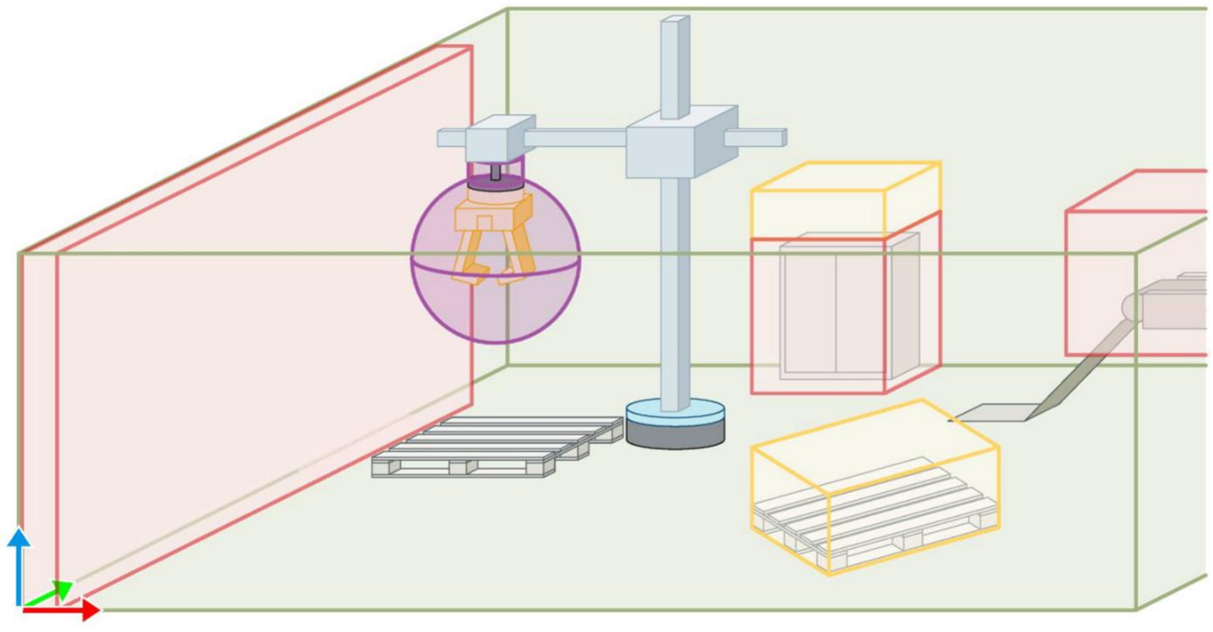


Figure 3.7 Example of zone placement in Technology Object

3.6.1 TIA Openness

One of the great features of the TIA Portal is that it provides a public API for interaction with the program. This API is called TIA Openness and using this API one can easily automate repetitive tasks in project development, generate new projects based on custom template, extract and import from existing projects and so on. The API supports all languages from the .NET Framework such as C#.

The user can run a program with or without a user interface based on his needs. The option without a user interface is faster to execute. The program development is straightforward. Several TIA Openness examples are available on the Siemens Industry Online Support page that can be used as a templates or inspiration for the task at hand. All the user needs to do, to use TIA Openness, is to install the TIA Portal and use the example project to create a simple application.

The only problem that might occurs is setting up the user rights to execute the TIA Openness application. The complete description is in the official manual [33], all the developer needs to do is to add the user account to the OpennessUser group, defined in the Windows operating system application called Computer Management.

3.7 NX MCD

NX is Siemens Computer Aided Design (CAD) software for construction design and modeling. NX Mechatronics Concept Designer (MCD) is an NX plugin for a simulation of the mechatronic behavior of the current 3D model. It enables the simulation of various movements of the parts of the 3D model and it is also possible to control these movements from outside the NX software. For example, it is possible to connect the variables that control speed of a rotor to PLCSIM Advanced for testing the developed PLC program. Mechatronics Concept Designer has easy-to-use modeling and simulation which allows you to quickly create and validate alternative design concepts early in the development cycle. Unlike a model-based tool, MCD allows you to not only see what it looks like but validate that it works. This concept of developing a software and a model fits in the new virtual workflow of commissioning.

3.7.1 NX Open

The NX software provides a public API for accessing functions within the software. This API is called NX Open and it supports several common programming languages. These include C/C++ , Java, Python and all .NET Framework languages.

The NX Open has several ways how to execute a developed program. The one used in this thesis is called the Command Line Method. This method can only run as batch mode, therefore there can be no interaction with the running GUI (Graphical User Interface) of NX. The specifics of this execution method will be discussed in the proposed solution section.

An easy way to create a script for automating a task or accessing data in NX is via the Journal feature. The Journal is like a macro. The user starts recording using the journal and the journal creates a record of what the user has done and exports it to a script into the desired programming language (i.e. C#). This script can then be used for creating a more complex automation task.

4 Kinematics

This chapter contains a kinematic description of an object in 3D space and methods for converting between the coordinate systems of the Unity and the TIA Portal. It describes different methods for describing a general rotation and the problems that arises with these descriptions.

4.1 Kinematic Description of an Object in 3D Space

An object in general 3D space has 6 degrees of freedom which can be described as a position in 3D space and a rotation in 3D space. To be able to define a position and rotation of an object in space a Coordinate System (CS) must be defined. In software engineering the most common is the Cartesian Coordinate System.

In 3D space a Cartesian Coordinate System is a set of 3 linear axes that are mutually perpendicular. The Cartesian Coordinate System is usually denoted as O(x,y,z) where O means the origin of the coordinate system and (x,y,z) are the coordinates. It is up to the developer to choose the orientation of these axes. For this reason, the Cartesian Coordinate Systems are divided into left-handed CS and right-handed CS based on the axis orientation.

4.1.1 Position

As mentioned before position is mostly described using the Cartesian Coordinate System O(x,y,z) as is displayed in the following Figure 4.1.

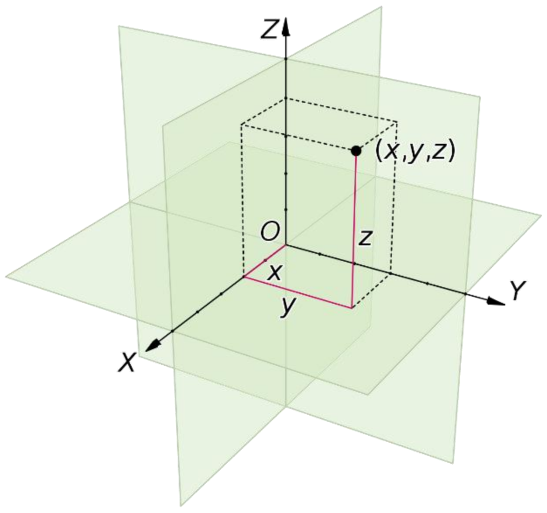


Figure 4.1 Cartesian Coordinate System

4.1.2 Rotation

A common and perceptible way of describing a rotation in 3D space are Euler Angles. Euler Angles take advantage of Euler's theorem which implies that any orientation can be achieved by composing three elemental rotations in which two consecutive rotations are not about the same axis. This condition is fulfilled for example by rotations about the axes of the Cartesian Coordinate System.

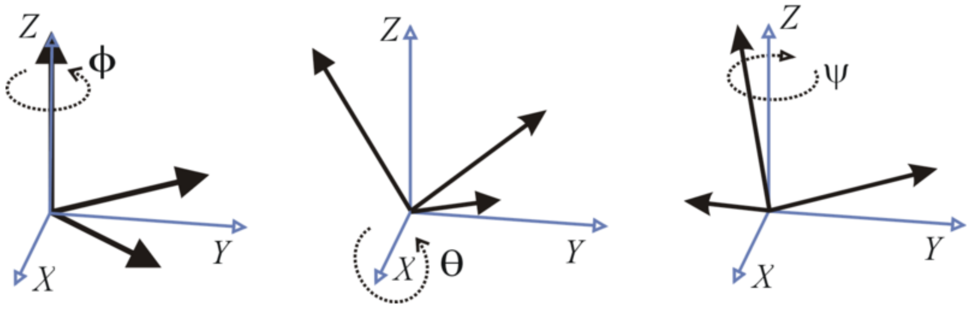


Figure 4.2 Euler Angles (ZXZ) fixed (extrinsic) rotation

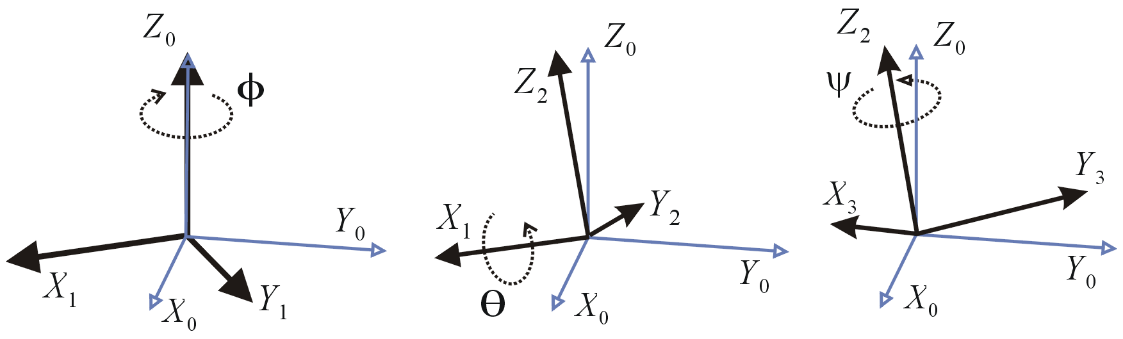


Figure 4.3 Euler Angles (ZXZ) mobile (intrinsic) rotation

There are many conventions when using Euler Angles because there are plenty of ways to compose the order of rotation around different cartesian axes. When using the cartesian axes, only twelve unique meaningful ordered sequences of rotations exist or twelve Euler Angle conventions: XYX, XYZ, XZX, XZY, YXY, YXZ, YZX, YZY, ZXY, ZXZ, ZYX, ZYZ [34].

The use of Euler Angles can be further divided into intrinsic and extrinsic conventions. When given a Euler Angle sequence convention (i.e. XYZ) you must specify whether you will rotate about fixed axes (extrinsic) or about newly emerged axes (intrinsic).

Because the rotation can be described with a minimum of 3 parameters, Euler Angles are a minimal representation of object orientation in space. Therefore, the parameters are independent of each other.

When programming 6DOF manipulator robots in the TIA Portal a convention called E6AXIS or E6POS is used. This is a common convention and is used by Siemens strategic partner KUKA which is a supplier of robotics and plant manufacturing systems [35].

The E6AXIS convention is a structure of 6 parameters describing axis rotation in degrees and 6 optional parameters describing external axes. The axis parameters are denoted A1, …, A6 and are visualized in Figure 4.4 . This figure is copied from the official user manual [36]. It is possible to see that the position of each axis rotation is dependent upon the previous axis position. Therefore, the E6AXIS convention is an intrinsic convention.

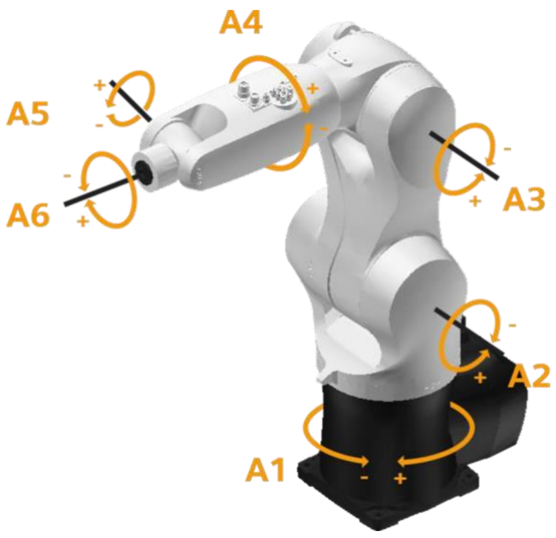


Figure 4.4 The E6AXIS structure

The E6POS is a structure of 6 + 2 parameters describing the position and rotation of the coordinate system of the robot's end position (i.e. the tool). The 2 parameters, called Status and Turn, describe a more precise specification of the alignment at the current position with Cartesian position specification.

The 6 parameters of the E6POS structure are usually denoted as X, Y, Z describing the position and A, B, C describing orientation. A is orientation around the Z axis, B the Y axis and C the X axis in that order [36]. The official documentation "S7-1500T Kinematics Functions V4.0 in TIA Portal V15" unfortunately does not explicitly say whether an intrinsic or extrinsic rotation is used for E6POS, so using the GUI in the TIA Portal it was shown that the TIA Portal uses intrinsic convention. The documentation probably expects that the convention in robotics is to use the intrinsic convention called yaw, pitch, roll.

Coordinates	**Description**
x	Shift in the x direction in the reference coordinate system
y	Shift in the y direction in the reference coordinate system
z	Shift in the z direction in the reference coordinate system
A	Rotation around the z-axis in degrees
B	Rotation around the y-axis in degrees
C	Rotation around the x-axis in degrees

Table 4-1 Coordinates in E6POS structure

Therefore, E6POS uses an intrinsic ZYX convention of Euler Angles. The coordinate system of the robot's endpoint is visualized in Figure 4.5.

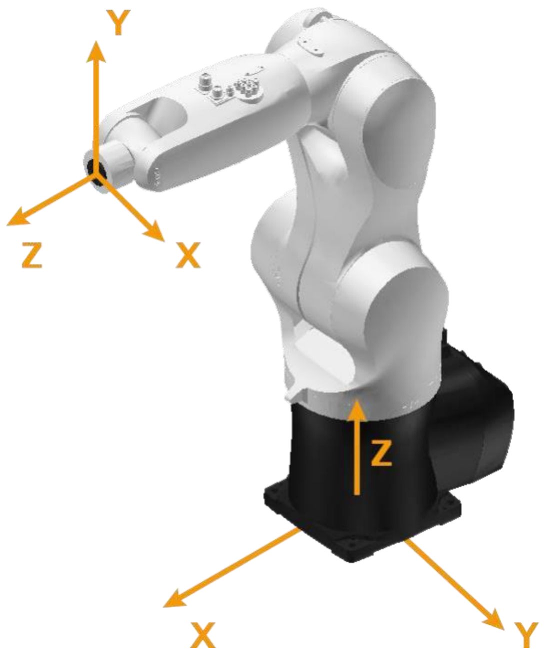


Figure 4.5 The E6POS structure

Significant disadvantage of using Euler Angles as a representation of general rotations is that they suffer from a phenomenon called Gimbal Lock. When applying the three rotations in turn, it is possible for the first or the second rotation that the third rotation axis points in the same direction as one of the two previous rotation axes. This means that a degree of freedom is lost because the third rotation cannot be applied around a unique axis. Therefore, any rotations are in degenerate two-dimensional space.

In practice this means that when Gimbal lock occurs there are infinite possibilities of how to represent current rotations using the 3 Euler Angles. This then leads to numerical instability even when the system is near this configuration because the parameters that are used to calculate the angles are approaching to infinity or zero. An easy solution to resolve this problem is to lock one angle value and calculate the other two, unfortunately this leads to discontinuous behavior which might be problem for follow-up calculations.

In the case of yaw, pitch, roll (ZYX) rotation, the gimbal lock occurs when pitch approaches ±90 degrees. To avoid this problem, quaternions are used represent general rotation in space.

4.1.3 Quaternion

The quaternions are an extension of complex cumbers. They are noncommutative and they can be defined as a set of 4 numbers with defined operations for adding and multiplying. The quaternion has one real and three imaginary components and is usually denoted either as an ordered pair of a real number and vector

$$q = (w, \vec{v}); \;\; w \in \mathbb{R}, \vec{v} = (x, y, z) \in \mathbb{R}^3 \tag{1}$$

or as a number

$$q = w + x\boldsymbol{i} + y\boldsymbol{j} + z\boldsymbol{k}; \quad w, x, y, z \in \mathbb{R} \tag{2}$$

where $\boldsymbol{i}, \boldsymbol{j}, \boldsymbol{k}$ denotes the imaginary components. The defined operations fulfill the following equations [37]

$$\begin{gathered} \boldsymbol{i}^2 = \boldsymbol{j}^2 = \boldsymbol{k}^2 = \boldsymbol{ijk} = -1 \\ \boldsymbol{ij} = \boldsymbol{k}, \qquad \boldsymbol{jk} = \boldsymbol{i}, \qquad \boldsymbol{ki} = \boldsymbol{j} \\ \boldsymbol{ji} = -\boldsymbol{k}, \qquad \boldsymbol{kj} = -\boldsymbol{i}, \qquad \boldsymbol{ik} = -\boldsymbol{j} \end{gathered} \tag{3}$$

The conjugated quaternion is denoted

$$\bar{q} = q = w - x\boldsymbol{i} - y\boldsymbol{j} - z\boldsymbol{k} \tag{4}$$

The quaternion can be used as a non-minimal representation of general rotation in 3D space. An intuitive way of understanding a quaternion as 3D rotation is to think that in the ordered pair $q = (w, \vec{v})$, $\vec{v}$ represents an axis in 3D space and w represents a rotation angle about this axis.

When using quaternion as general representation of rotation, a unit quaternion is used. Therefore, the following restriction must be applied.

$$|q| = \sqrt{w^2 + x^2 + y^2 + z^2} = 1 \tag{5}$$

Quaternion can then be used as a rotation operator of 3D vector $\vec{p}$ the following way.

Suppose we want to rotate a 3D vector $\vec{p}$ around axis $\vec{v} = [v_1, v_1, v_1]$ about θ degrees. The quaternion is calculated as

$$q = \cos\frac{\theta}{2} + \sin\frac{\theta}{2}(v_1\boldsymbol{i} + v_2\boldsymbol{j} + v_3\boldsymbol{k}) \tag{6}$$

The vector $\vec{p}$ must be written as pure quaternion

$$\vec{p}_q = p_1\boldsymbol{i} + p_2\boldsymbol{j} + p_3\boldsymbol{k} \tag{7}$$

The desired rotated vector is calculated as

$$\vec{p}_q' = q \cdot \vec{p}_q \cdot \bar{q} \tag{8}$$

4.2 Converting Rotations Between Unity and TIA Portal

Unity software uses left-handed CS while the Siemens's NX MCD and TIA Portal uses right-handed CS. The following table describes the differences in coordinate systems from Unity software to other software solutions. The direction is considered from the object's point of view when looking forward.

Direction	Unity	TIA Portal
Forward	z	-y
Right	x	-x
Up	y	z

Table 4-2 Mapping axis from Unity to TIA Portal coordinate system

The differences between Unity and TIA Portal can be seen on GUI. Below is a comparison of a workspace coordinate system in Unity and in TIA Portal.

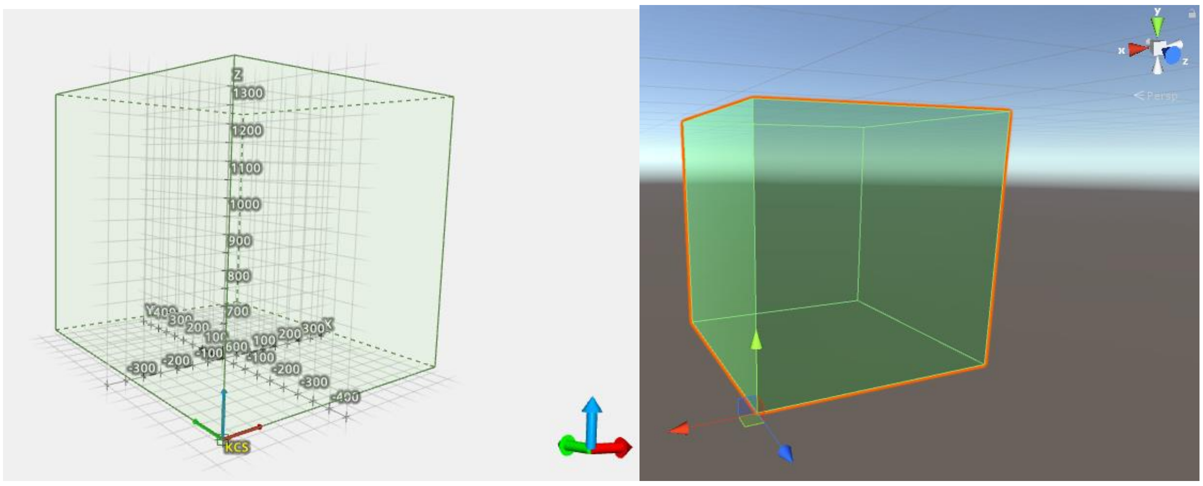


Figure 4.6 Coordinate system orientation comparison of TIA Portal (left) and Unity (right)

4.2.1 Converting rotation from TIA Portal to Unity

Using the Figure 4.6 it is possible to calculate the transformation of TIA Portal coordinates to Unity coordinates of workspace zone. Thanks to Unity API, the conversion can be easily done. The following code demonstrates positioning of workspace zone in Unity using TIA Portal coordinates E6POS.

```
//position
Vector3 position = new Vector3(x, y, z);
position.x = - position.x;
position = Quaternion.AngleAxis(270, Vector3.right) * position;
// rotation
Quaternion rotation =
        Quaternion.AngleAxis(-A, Vector3.up) *
        Quaternion.AngleAxis(B, Vector3.forward) *
        Quaternion.AngleAxis(C, Vector3.right);
```

Snippet 1 Transforming coordinates from TIA Portal to Unity

The previous code example first negates the coordinate of x and then rotates the vector around Unity’s x-axis about 270 degrees. This vector then represents the position of the cube’s pivot point in space. Then the cube’s rotation is calculated using Unity API. The method `Quaternion.AngleAxis(angle, axis)` calculates quaternion that represent rotation around the vector axis and angle in degrees.

The rotation of the cube can be calculated using the property that quaternions have in Unity. Rotating a vector by the product of two quaternions is the same as applying the two rotations in sequence – first you apply the left-hand side quaternion and then you apply the right-hand side quaternion relative to the reference frame resulting from left hand side rotation [20].

It is important to note that the angle at parameter A is negated. This is because we are changing from a right-handed to left-handed coordinate system. Therefore, we must check each axis and use the left-hand rule to determine the correct positive direction of rotation and compare it with the original.

The positive direction of a coordinate system can be easily determined by using the right-handed rule for right-handed CS and the left-handed rule for left-handed CS. To determine positive direction, point the correct hand's thumb in the direction of rotational axis. The direction in which your fingers curl and point is the direction of positive rotation.

4.2.2 Converting Rotation from Unity to TIA Portal

Calculating the rotation of a cube in Unity back to TIA coordinates is more difficult because of the conversion to Euler Angles. They, as mentioned before, suffer from singularities due to the gimbal lock which can lead to numerical instability.

When given a quaternion in Unity coordinates, the general conversion process to different CS is to map the quaternion axis into the new coordinate system and if changing from left to right-coordinate systems, negate the rotational angle [38]. Compare Table 4-2 and the implementation in the following Snippet 2 example.

```
Quaternion ConvertToTIA(Quaternion input){
       // map the axis
       Quaternion q = Quaternion.identity;
       q.x = -input.x;
       q.y = -input.z;
       q.z =  input.y;
       q.w =  input.w; // don't map

       // negate angle
       q.x = -q.x;
       q.y = -q.y;
       q.z = -q.z;
       return q;
}
```

Snippet 2 Convert Unity Quaternion to TIA coordinates

At first, the function converts the Unity quaternion to TIA Portal quaternion and then use the following set of equations

$$\begin{bmatrix}\phi\\ \theta\\ \psi\end{bmatrix} = \begin{bmatrix}\tan^{-1}\dfrac{2(q_w q_x + q_y q_z)}{1-2(q_x^2+q_y^2)}\\ \sin^{-1}2(q_w q_y - q_z q_x)\\ \tan^{-1}\dfrac{2(q_w q_z + q_x q_y)}{1-2(q_y^2+q_z^2)}\end{bmatrix} \tag{9}$$

to convert from quaterion to euler angles int ZYX convention. These equations were taken from a report by José Luis Blanco Claraco [39].

5 Proposed Solution

The proposed solution visualizes these zones with the 3D model of the robot, which was previously created in NX MCD, in Microsoft HoloLens. It will also visualize these zones in NX MCD software so that the user will have multiple ways of visualizing these zones. In order to improve the workflow of commission it is proposed to take advantage of a HoloLens headset and visualize the workspace zones of industrial robots defined in the TIA Portal.

There are three types of workspace zones. The first one is Work Zone. This is where the robot manipulator can operate. The second one is Signal Zone, which is a zone within Work Zone where the software will generate a signal if the robot enters this zone. The last one is Blocked Zone. This zone does not necessarily have to be within Work Zone, but it usually is because there is no point of defining Block Zone outside Work Zone since the robot should not escape the Work Zone.

The solution is divided into two parts. The backend server that runs on a PC and a frontend HoloLens application called HoloSee that communicates with the PC over Wi-Fi using TCP Sockets.

5.1 Backend Server Application

The backend application is divided into five different projects and one standalone project for automating tasks in Unity editor.

The five projects in the backend server application are listed in the table below with a brief description.

Project name	Job
SimpTiaAPI	Simplified API for communicating with TIA portal using TIA Openness
NxBuilder_run_managed	Simplified API for creating and deleting features in NX MCD using NX Open
Convert3DModelToAssetBundle	An API for creating Asset Bundles from .dae files using Unity editor
SynchronousSocketServer	A server that listens at specified IP and port and responds to any incoming request. Provides Asset Bundle and XML data structure of work zones

ServerApplicationForHololens	Manages the application algorithm
Standalone **ImportAssetBundle**	Unity project for debugging C# script used in **Convert3DModelToAssetBundle** project

Table 5-1 List of projects in Server Application

5.1.1 Import Asset Bundle Project

The Import Asset Bundle is a standalone project for debugging a script `ImportModelAndCreateBundleClass.cs`, which is then used in the main application. It is an empty unity project where only one asset/script is present. The script uses Unity editor scripting API and standard C# libraries for creating an Asset Bundle. According to official documentation, every script that is meant to extend the features of Unity editor must be in a folder called "Editor" somewhere in the asset folder. An easy way to launch a script within Unity editor is to create a static void method with an attribute `[MenuItem("MyMenu/DoSomething")]`. This will create a menu item in the editor menu bar and when pressed it executes this method.

The script reads the program arguments using the `System.Environment.GetCommandLineArgs()` method. This includes specifying the path to .dae model, output path for Asset Bundle and Asset Bundle name. The script then sets up the project's Player Settings. This means defining scripting backend and a scripting runtime version. Afterwards, it sets up build settings for the UWP HoloLens application. After setting up the project settings it copies the .dae file to asset folder and refreshes the asset database using the static method `AssetDatabase.Refresh(ImportAssetOptions.Default)`. Then it creates the desired Asset Bundle in the desired directory within the asset folder.

The Complete algorithm is shown on the following Figure 5.1.

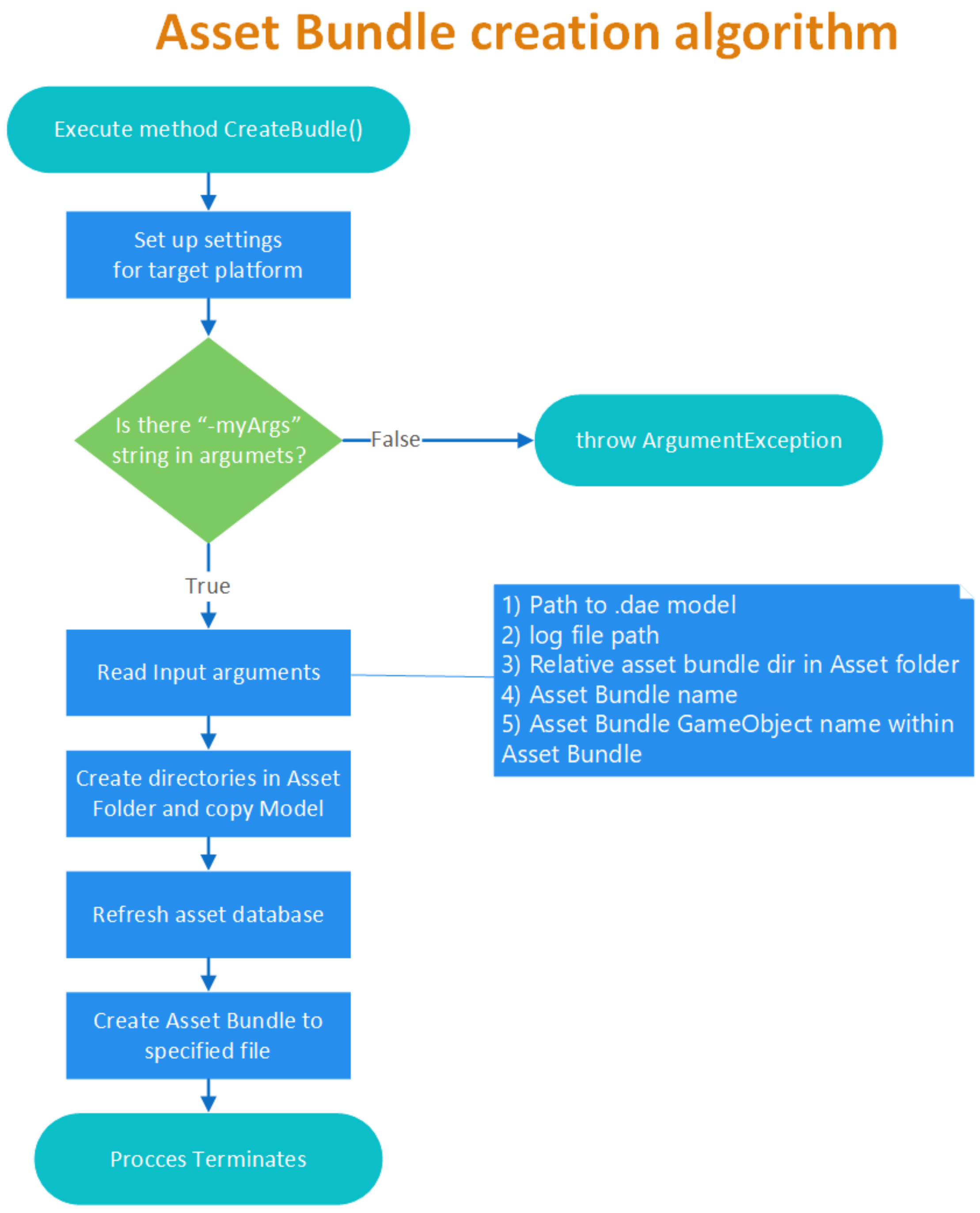

Asset Bundle creation algorithm
Execute method CreateBudle()
Set up settings for target platform
Is there "-myArgs" string in argumets?
False
throw ArgumentException
True
Read Input arguments
1) Path to .dae model
2) log file path
3) Relative asset bundle dir in Asset folder
4) Asset Bundle name
5) Asset Bundle GameObject name within Asset Bundle
Create directories in Asset Folder and copy Model
Refresh asset database
Create Asset Bundle to specified file
Procces Terminates

Figure 5.1 Asset Bundle creation algorithm

5.1.2 Simplified TIA API project

The first project uses TIA Portal Openness to access information within a TIA Portal project. The user must give the program access to the TIA Portal at the beginning of the execution which is done in the TIA Portal popup window. If the user selects "Yes to all" he will not have to allow it again during the process lifetime. The project consists of three classes which are then called from the application algorithm. The UML domain diagram is visualized in Figure 5.2.

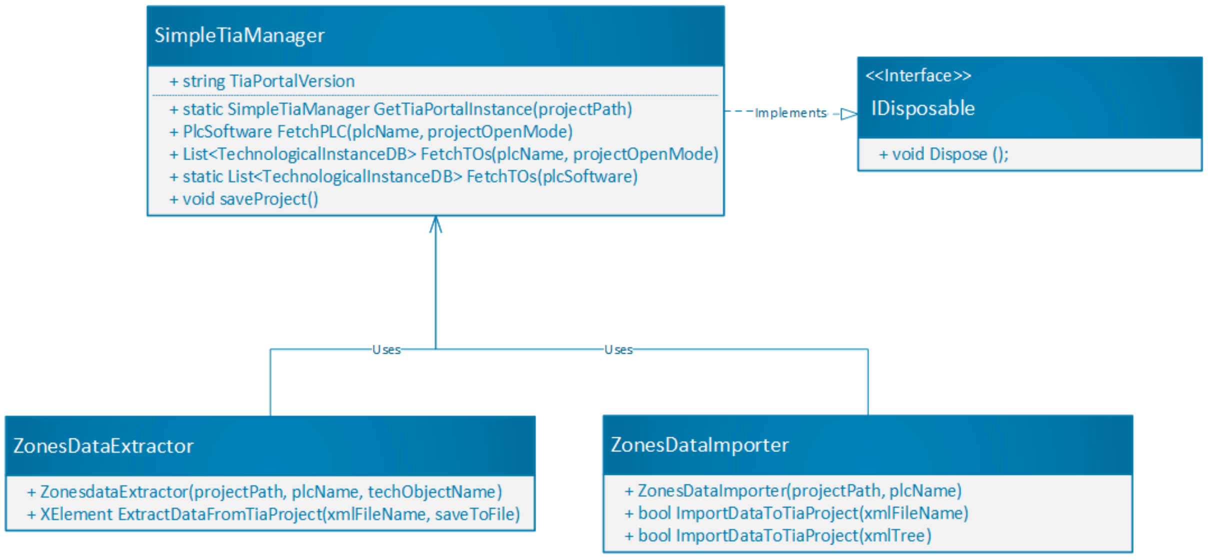

SimpleTiaManager
+ string TiaPortalVersion
+ static SimpleTiaManager GetTiaPortalInstance(projectPath)
+ PlcSoftware FetchPLC(plcName, projectOpenMode)
+ List<TechnologicalInstanceDB> FetchTOs(plcName, projectOpenMode)
+ static List<TechnologicalInstanceDB> FetchTOs(plcSoftware)
+ void saveProject()
Implements
<<Interface>>
IDisposable
+ void Dispose ();
Uses
Uses
ZonesDataExtractor
+ ZonesdataExtractor(projectPath, plcName, techObjectName)
+ XElement ExtractDataFromTiaProject(xmlFileName, saveToFile)
ZonesDataImporter
+ ZonesDataImporter(projectPath, plcName)
+ bool ImportDataToTiaProject(xmlFileName)
+ bool ImportDataToTiaProject(xmlTree)

Figure 5.2 UML Domain diagram of SimpleTiaAPI project

The `SimpleTiaManager` class provides a simplified interface of TIA Portal Openness for extracting and importing elements of data from the TIA Portal project. The `ZoneDataExtractor` class is created using the constructor. The constructor arguments are a string path to the saved project, string name of the PLC within the project, and Technological Object name of type 'TO_Kinematics'within the PLC in the project. The data extracted from the TIA Portal consist of a list of work zones within the specified technological object. An example code can be found in the Figure 13.1 in the appendix.

This project was developed using the application example available at Siemens Online Industry Support (SIOS) [33] where a pdf and a Visual Studio solution is available. The pdf describes the demo application and serves as a documentation for TIA Openness. In order to compile the demo application, the user must reference Siemens.Engineering.dll and Siemens.Engineering.HMI.dll to Visual Studio solution. These files are in the installation directory of the TIA Portal. No major issues or problems were encountered during the development of the Simplified TIA API project. The manual is very well written.

The hardest task during the development was to find the desired data within the TIA Project. Fortunately, the pdf manual contains a class diagram of all classes and their inheritance which was very helpful. Another key thing is to realize that the classes and data structures correspond with the GUI of the TIA Portal and therefore it is crucial to find the data in GUI and after that, using the class diagram, to find the desired data.

Below is a figure with the class diagram for PLCs components in TIA Portal.

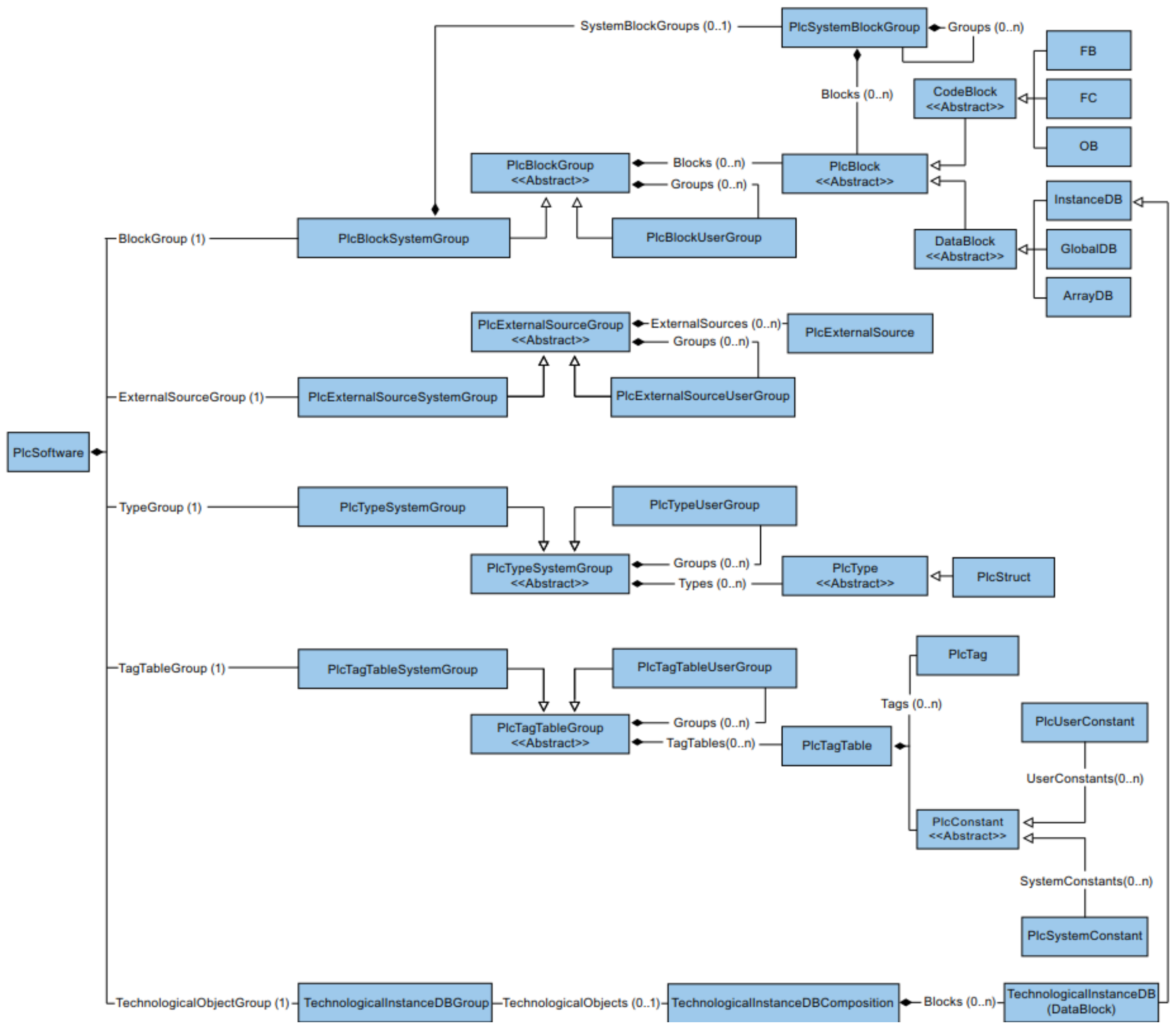


Figure 5.3 Partial class diagram of TIA Openness project [40]

5.1.3 The NX Builder Project

The NX Builder project is composed of two classes `ZonesBuilder.cs` and `Program.cs`. These classes are compiled to exe file and then launched as a separate process from the main application program. The NX Builder program is started as a separate process because of how NX Open executes batch mode applications. The official documentation for NX Open lists the following options to execute batch applications using the command line [41]:

An NX Open application .exe file can be executed directly from a command line as any other executable. Since this is a managed application, you will need to do one of the following:

1) Copy the NX .NET libraries to your local working directory. To do so, copy all of the libraries from the %UGII_BASE_DIR%\ugii\managed directory to your working directory. Use standard operating system command to execute the application.

2) Copy your .EXE to %UGII_BASE_DIR%\ugii\managed. Use standard operating system command to execute the application.
3) Use *run_managed.exe* (%UGII_BASE_DIR%\ugii\run_managed.exe)

run_managed.exe is a standalone executable that runs a managed NXOpen .EXE in the correct environment allowing it to pick up other DLLs from the install when they are not in the same directory as the .EXE itself.

usage:

```
run_managed <executable-file> <arguments>
```

Since it is desired to distribute the software solution separately from the NX MCD and TIA Portal the most convenient solution, at least from the end-user point of view, is to use the third option. This way the user does not have to copy or alter any files within the installation directory of NX in any way, so there is a smaller chance that something goes wrong. Another issue could be that the user does not have administrator rights to the installation directory. Therefore, he would not be able to change anything in the directory and the application would not work. When using *run_managed.exe*, the user only has to provide a path to *run_managed.exe* which resides in the installation directory of NX MCD.

The main application algorithm first creates an Anonymous pipe using the class `AnonymousPipeServerStream` then it creates a sub-process with run_managed.exe using the `Process` class in .NET Framework and gives the process a handle to the created `AnonymousPipeServerStream` class instance as an argument. The sub-process then reads data from the pipe and executes one of the commands specified in the enumeration `Cmds`. These commands are received via the pipe. The communication and sub-process execution are visualized in Figure 5.4.

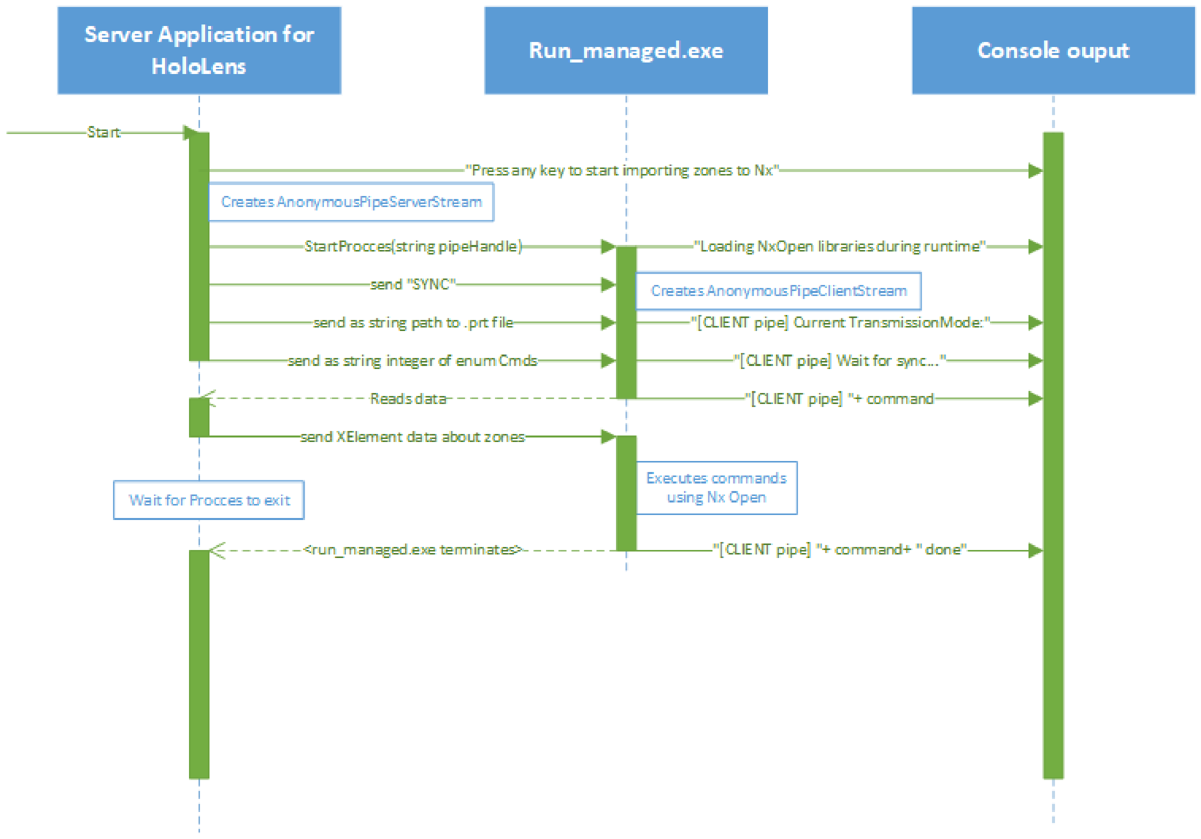

Server Application for HoloLens
Run_managed.exe
Console ouput
Start
"Press any key to start importing zones to Nx"
Creates AnonymousPipeServerStream
StartProcces(string pipeHandle)
"Loading NxOpen libraries during runtime"
send "SYNC"
Creates AnonymousPipeClientStream
send as string path to .prt file
"[CLIENT pipe] Current TransmissionMode:"
send as string integer of enum Cmds
"[CLIENT pipe] Wait for sync..."
Reads data
"[CLIENT pipe] "+ command
send XElement data about zones
Executes commands using Nx Open
Wait for Procces to exit
<run_managed.exe terminates>
"[CLIENT pipe] "+ command+ " done"

Figure 5.4 NX Open execution

5.1.4 Convert 3D Model to Asset Bundle Project

The conversion from .dae model to an Asset Bundle is done by using Unity editor in batch mode. There are several arguments that can be used when using a batch mode, the complete list can be found in the official Unity Manual [19]. The ones used here are:

```
-createProject <pathname>
-projectPath <pathname>
-executeMethod <ClassName.MethodName>
```

The first command line argument creates an empty project to a specified path. The second command opens a project in a specified path. The third command line argument executes a static method of a scripts that must reside within an editor folder.

The algorithm goes like this:

1. Read input data arguments. These arguments include Unity editor exe path, .dae model path, project path for new empty project, Asset Bundle path etc.
2. Start new Unity editor process in batch mode, create empty project and wait for process to exit

3. Copy the embedded `ImportModelAndCreateBundleClass.cs` from assembly resources to editor folder under the asset folder.
4. Start new Unity editor process in batch mode, open the project and execute the method within the `ImportModelAndCreateBundleClass.cs` script. The script executes as described in chapter 5.1.1.
5. The algorithm exists

The main application algorithm now has the Asset Bundle in the specified location.

5.1.5 Synchronous Socket Server

The synchronous socket server project uses .NET Framework implementation of Sockets and listens to a specified endpoint. The project uses only one class called `SynchronousSocketListener`. This class has a public constructor with four parameters – IP address, port number, path to Asset Bundle and XML tree representing the workspace zones.

After the class instantiation the user calls public method `StartListening(…)` which starts the server application. The application then listens for incoming requests for one second. If there are none it then checks if the user pressed the "Q" button. If so, the application terminates; otherwise it continues in a loop and listens for requests for another second.

In the `SynchronousSocketListener` class there are three different enumeration types used for communication.

```
public enum CommunicationToServer {NoComCmd, ServerCanCloseConnection,
TransferToClientCompleted}
public enum CommunicationToClient {ClientCanCloseConnection,
ransferToServerCompleted}
public enum CommandsToServer {NoCmd, TestConnection, SendMeXML, UpdateXml,

SendMeRobotAssetBundle}
```

Snippet 3 Enumeration of commands for communication

The `CommunicationToServer` enumeration represents messages that the client sends to the server for confirmation of a response. The `CommunicationToClient` enumeration represents messages that the server sends to a client in response to the `CommunicationToServer`.

`CommandsToServer` represents a list of commands from the client that the server is supposed to do. The `TestConnection` is used whenever a connection is initialized and is used mainly for debugging. The `SendMeXML` command tells the server to send an XML data describing the workspace via socket. The `UpdateXml` tells the server to listen for incoming data on the socket representing the new XML data describing the workspace. The last command, `SendMeRobotAssetBundle` tells the server to send an Asset Bundle that was created using Unity editor.

The communication that happens between the server and client when the client requests XML data about workspace zones and Asset Bundle holding the 3D model is visualized in the following figure.

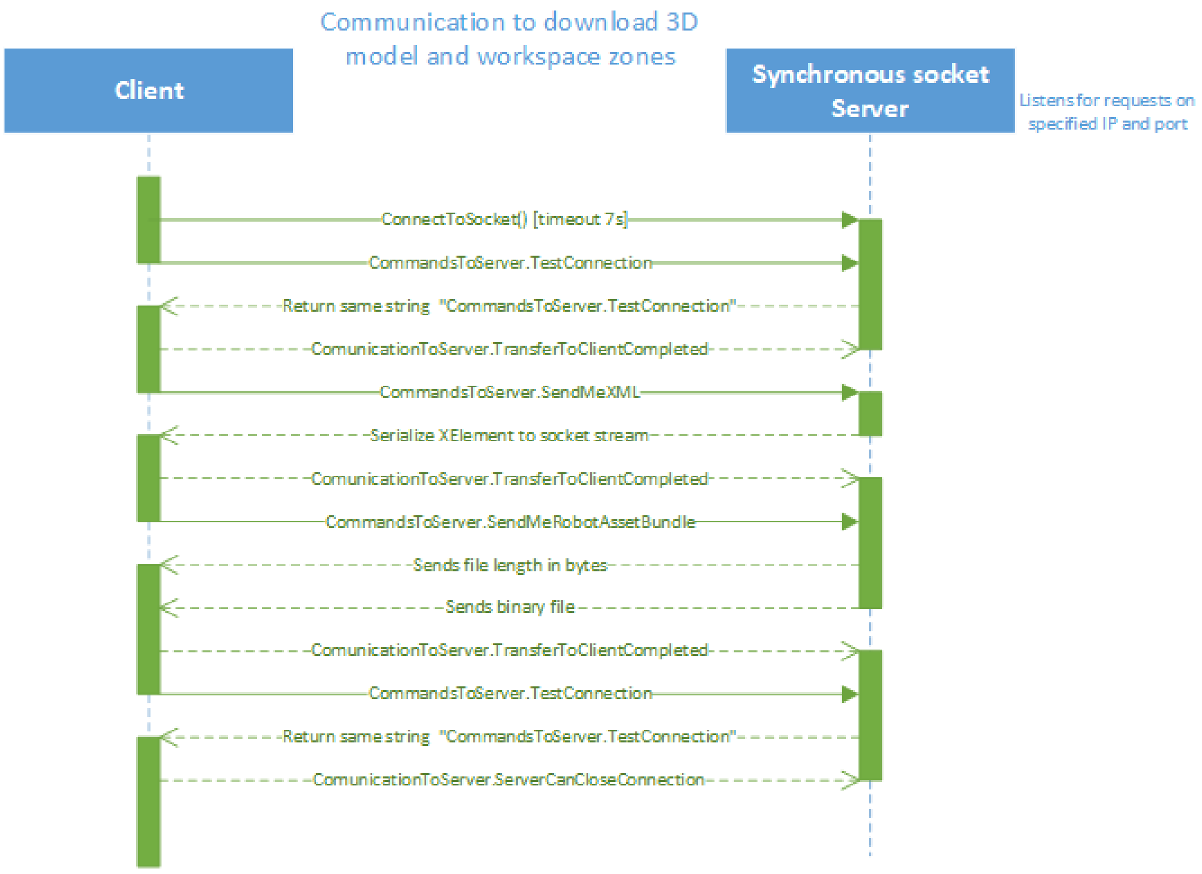

Communication to download 3D model and workspace zones
Client
Synchronous socket Server
Listens for requests on specified IP and port
ConnectToSocket() [timeout 7s]
CommandsToServer.TestConnection
Return same string "CommandsToServer.TestConnection"
ComunicationToServer.TransferToClientCompleted
CommandsToServer.SendMeXML
Serialize XElement to socket stream
ComunicationToServer.TransferToClientCompleted
CommandsToServer.SendMeRobotAssetBundle
Sends file length in bytes
Sends binary file
ComunicationToServer.TransferToClientCompleted
CommandsToServer.TestConnection
Return same string "CommandsToServer.TestConnection"
ComunicationToServer.ServerCanCloseConnection

Figure 5.5 Communication process between client and server when downloading data from server

5.1.6 Server Application for HoloLens

The Server Application for HoloLens is a project that binds all projects together and controls the algorithm of the whole application. Figure 5.5 visually describes the application algorithm. First, the application reads input arguments. If the application finds that there is at least one argument available, it interprets the first argument as a path to an XML file where the server settings are stored. Otherwise it will look for XML file settings within the current directory. Then the application verifies the settings using static method `verifySettings(…)`.

After that, the application tries to find the TIA Portal session, or it will create a new one using the Simple TIA API project. With this session, the app extracts data about workspace zones.

Then it deletes any created zones in the specified .prt file using the NX Builder project. If the user specified that he wishes to the export model from NX the algorithm proceeds to export .prt file to .dae. If the user specified otherwise, the user has to do the export manually and specify a path to the exported .dae file. Then the algorithm recreates the zones again using NX Builder project. After the zone’s creation in .prt file, the algorithm starts the Convert 3D Model to Asset Bundle project, which will create the Asset Bundle.

After all this is done, the application starts acting as a server and starts to listen to the port and IP address specified in the server settings. The algorithm uses the last project Synchronous Socket Server and communicates with the client as described in the previous section.

The complete algorithm for the server application for HoloLens is visualized on Figure 5.6.

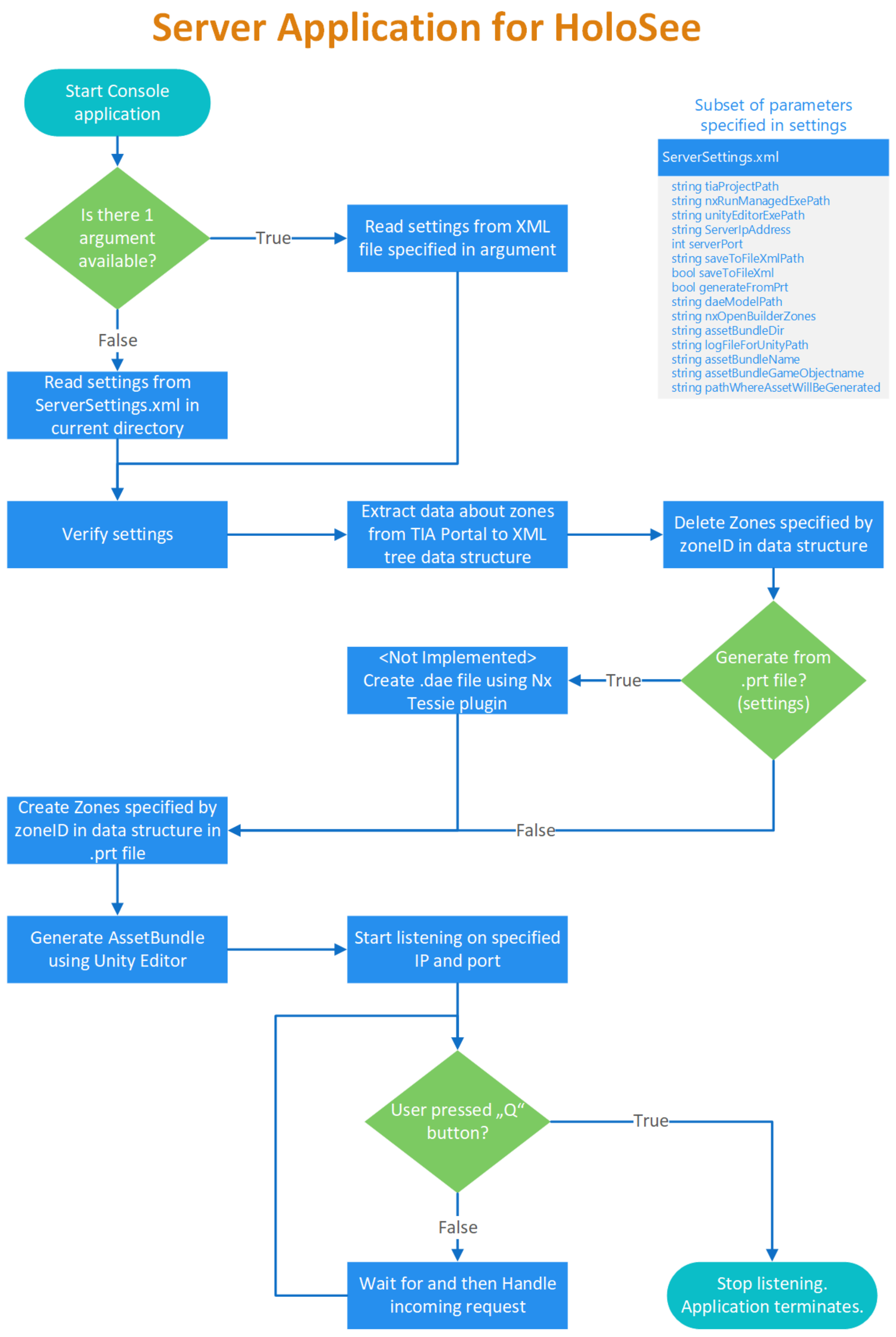

Server Application for HoloSee
Start Console application
Is there 1 argument available?
True
Read settings from XML file specified in argument
False
Read settings from ServerSettings.xml in current directory
Subset of parameters specified in settings
ServerSettings.xml
string tiaProjectPath
string nxRunManagedExePath
string unityEditorExePath
string ServerIpAddress
int serverPort
string saveToFileXmlPath
bool saveToFileXml
bool generateFromPrt
string daeModelPath
string nxOpenBuilderZones
string assetBundleDir
string logFileForUnityPath
string assetBundleName
string assetBundleGameObjectname
string pathWhereAssetWillBeGenerated
Verify settings
Extract data about zones from TIA Portal to XML tree data structure
Delete Zones specified by zoneID in data structure
Generate from .prt file? (settings)
True
<Not Implemented> Create .dae file using Nx Tessie plugin
False
Create Zones specified by zoneID in data structure in .prt file
Generate AssetBundle using Unity Editor
Start listening on specified IP and port
User pressed „Q“ button?
True
False
Wait for and then Handle incoming request
Stop listening. Application terminates.

Figure 5.6 Main application algorithm

5.1.6.1 Exporting NX Model to Digital Asset Exchange (.dae)

The exporting of a 3D model from NX to Unity was a difficult task because nowadays the NX does not offer a fully compatible format for Unity.

The first idea was to export NX's .prt model to AutoCAD's .dxf model which NX offers. Unfortunately, this does not give a usable result since there were no materials included and the exported model lacks the correct rotations of its parts.

The second idea was to use a third-party software called PiXYZ [42]. This is a specialized software for exporting 3D models from various CAD software. This export gives good results, but it is a third-party software which had to be bought.

Fortunately, the Siemens research team in Nuremberg is currently working on a new feature for NX that will export 3D models to various 3D formats including digital asset exchange format (.dae). DAE is a publicly available specification for 3D interchangeable format adopted by standard ISO/PAS 17506:2012 [43]. A pre-release version of this feature was obtained from the department and integrated into the algorithm. The exported .dae format gives great results and is fully compatible with Unity.

5.2 Frontend HoloLens Application

The frontend HoloLens application has only one scene defined in Unity and this scene takes care of every aspect of the application. The application design is based on the user case specification described in chapter 2.1 and it is also based on the necessary frameworks recommended hierarchy structure for application development like HoloToolkit or Vuforia. The application scene hierarchy is visualized in the following figure

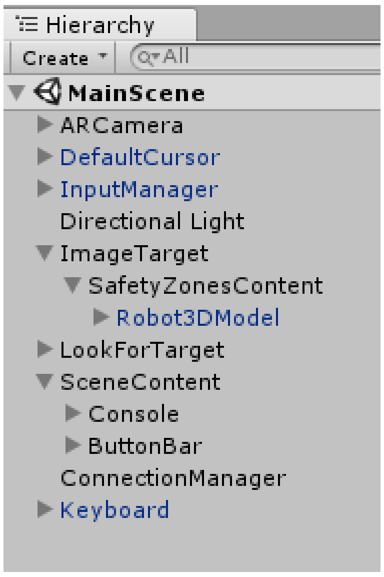


Figure 5.7 Scene hierarchy

AR Camera

The first element of the hierarchy is the augmented reality camera. This is a Unity regular camera game object with Vuforia behavior script attached. The Vuforia needs to be configured in Vuforia configuration file which is created when importing ARCamera before the application can use Vuforia's image recognition capabilities.

Default Cursor

The second element is the default cursor prefab that is provided with HoloToolkit.

Input Manager

The third element is Input Manager that takes care of user input. This means it detects user input like tap or tap&hold and notifies appropriate interaction receivers. The Input Manager uses physics raycast to detect where the user is looking so any interaction receiver must have a collider component attached in order to be able to receive any input from Input Manager.

Image Target

The fourth element is Image Target which is Vuforia's prefab for detecting images in the scenes. It has Image Target Behavior script component attached where the developer defines the target image. The image must be compiled using the Vuforia online service. The developer must upload the image to the server and then download the compiled version as a Unity package. Vuforia also offers an online service for compiling images during runtime but that is not used in this application. It is important to note that before the compiled image could be used, it must be enabled in Vuforia configuration.

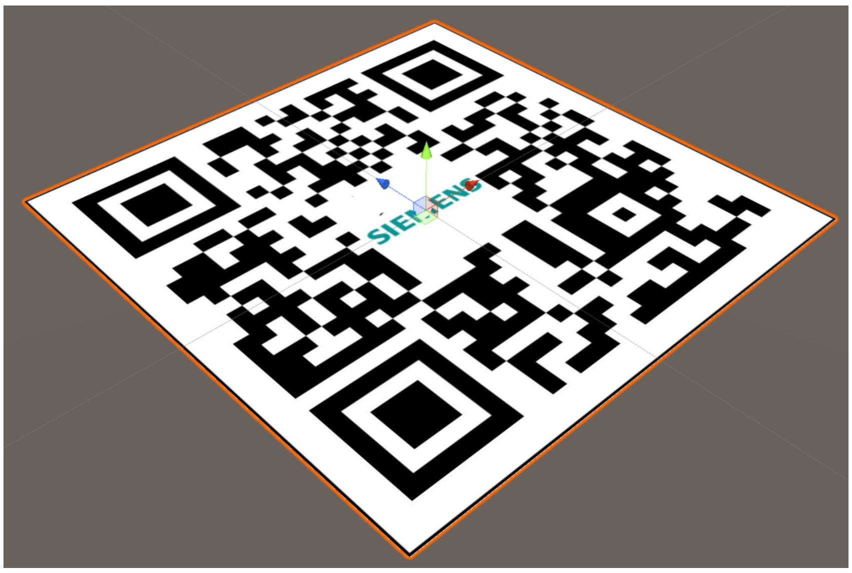


Figure 5.8 Image Target as it appears in Unity editor

Look for Target

The fifth element is Look for Target. This is a game object that appears at the beginning of the application to indicate to the user that he is supposed to look for a target image. After the image is found, this game object is destroyed using script that is attached to image target. The script contains a class that implements `ITrackableEventHandler` interface and receives a notification when the target image was found.

Look for Target Image

Figure 5.9 Indicator for a user that appears the beginning of the application

Scene Content

The sixth element is Scene Content game object. This game object represents the interface for communication between the application and the user as well as the communication between the user and the server. It contains a menu bar that has 5 buttons. The first button hides the console, with the exception of the hide button. The second button is showed when the hide button is pressed again. This button originally had a purpose but during development the purpose was removed, and it now acts similarly to the hide button. Pressing the show console will reveal the whole console. The third button will show/hide the imported 3D model so that the user can only see the work space zones. The fourth button executes method `StartClientCommunication()` in `ConnectionManager` which will be described later in this section. The fifth button executes method `UpdateXMLToSever()` in `ConnectionManager` and will be discussed later as well.

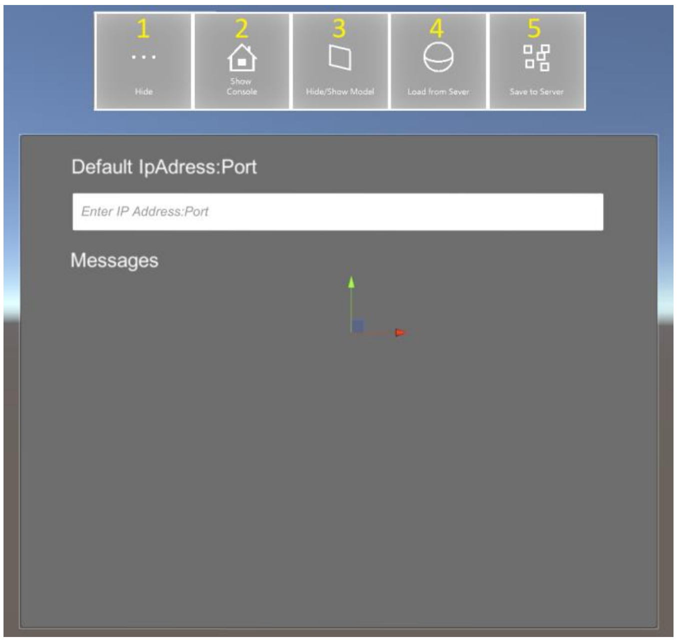


Figure 5.10 Console application for user interaction

The Scene Content game object also contains console game object. This game object contains an output field, where the messages from the application are written, and an input field where the IP address and port number are written and can be changed. Any changes in the input field are saved to a local XML file when the user presses the LoadFromServer button.

Connection Manager

The seventh element is Connection Manager. This game object has a component `SocketManagerHolder` class that has a private field with an instance of a class `SimpleStreamSocket`. This class takes care of all communication between the application and the server. This class communicates directly with the class `SynchronousSocketListener` described in chapter 5.1.6. The class contains an asynchronous method `StartComunication(Commands[])` with an input field of commands that the app wants to execute.

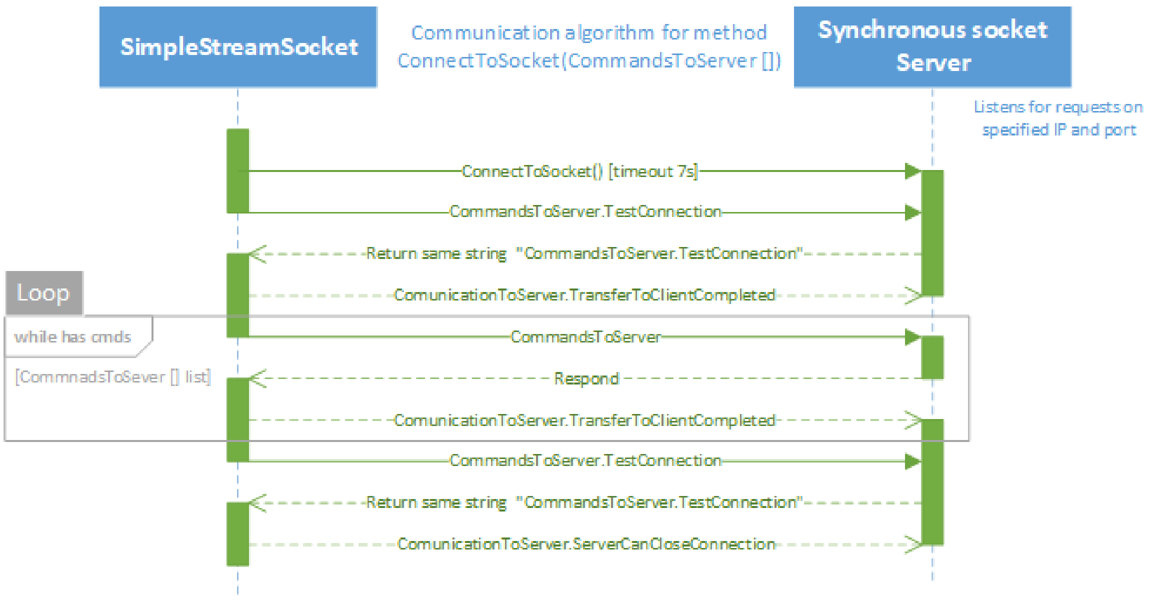

SimpleStreamSocket
Communication algorithm for method ConnectToSocket(CommandsToServer [])
Synchronous socket Server
Listens for requests on specified IP and port
ConnectToSocket() [timeout 7s]
CommandsToServer.TestConnection
Return same string "CommandsToServer.TestConnection"
ComunicationToServer.TransferToClientCompleted
Loop
while has cmds
[CommnadsToSever [] list]
CommandsToServer
Respond
ComunicationToServer.TransferToClientCompleted
CommandsToServer.TestConnection
Return same string "CommandsToServer.TestConnection"
ComunicationToServer.ServerCanCloseConnection

Figure 5.11 Communication between the HoloLens application and the Server

The `SimpleStreamSocket` has another method called `UpdateXMLToServer(XElement data)` which stores the `XElement` data to a private field of the instance and calls the `StartCommunication(…)` method with one command `UpdateXML`. The algorithm then sends the data in the private field to the server. The following Table 5-2 contains a list of the possible commands to the server.

Command name	Description
`TestConnection`	Command used for debugging and testing
`SendMeRobotAssetBundle`	Reads data from socket and stores the data as file in application persistent storage.
`SendMeXML`	Reads data from socket and stores the XML to `XElement` field
`UpdateXml`	Sends data stored in private field `XElement` of the class to the server.

Table 5-2 List of commands that can be send to server

Zone Prefab

Another element that is not in the hierarchy is the work zone prefab that is created during runtime based on the incoming data. The prefab's name is "zoneID" but this paper refers to this prefab as work zone prefab. The work zone prefab is a Unity's primitive cube type with `BoundingBoxScript`, `HandDragable` and `ColorDuringDraggingController` components attached. The prefab has a child game object that represents the primitive object because it is desired to have a pivot point on the same position as in the TIA Portal (right, front, down corner from the objects perspective). The `BoundingBoxScript` component creates menu bar and Bounding box that surrounds every work zone prefab. The menu bar consists of three buttons hide, adjust and move. As the name suggests, the hide button hides the menu bar and the work zone, adjust actives a box that is used for resizing the cube and move will allow the user to move the box by using tap & hold command.

The `BoundingBoxScript` and `HandDragable` are parts of the HoloToolkit framework and the `ColorDuringDraggingController` component has only one task; to make sure that during the moving of the work space prefab the color of the cube is changed to an opaque magenta color.

Keyboard

The keyboard is a part of HoloToolkit framework. It is a singleton class meaning only one instance or none is present during runtime. It takes care of user input when typing to the input field in scene content game object.

5.2.1 Encountered Problems During HoloLens Application Development

During the development several unexpected problems occurred. This section describes these problems and how they were solved.

5.2.1.1 Slow developments process

The first problem encountered was the slow development iteration time. This means the time needed to deploy an app from Unity to HoloLens device was too long for a smooth development. The time needed for a fresh new building and deployment of this application to HoloLens takes about 3 minutes 25 seconds.

The building process of exporting from Unity to Visual Studio solution takes about 1 minute 49 seconds. Opening the generated Visual Studio solution takes about 13 seconds. Building the solution in Visual Studio takes about 1:02 seconds and finally, deploying to HoloLens via USB cable takes about 21 seconds. The building process speeds up when rebuilding the same solution to the same folder, but it is still very slow.

In order to speed up the process, the HoloToolkit offers a simple debugging process using Unity play mode. The HoloToolkit offers a command in the menu bar "Mixed Reality Toolkit → Configure → Apply mixed reality Scene settings" which will import MixedRealityCameraParent. Using this camera game object, it is possible to move and use the tap command within the Unity editor play mode. This rapidly improves the iteration time.

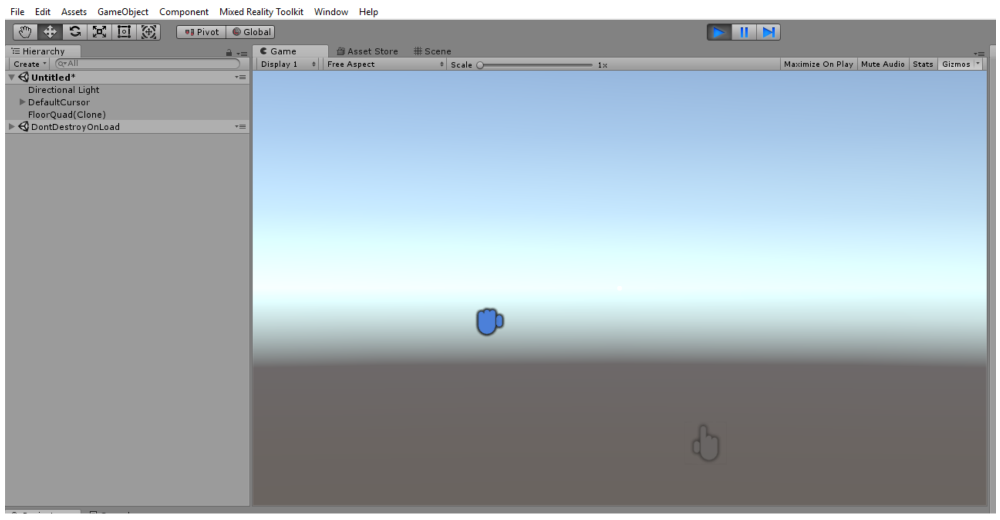
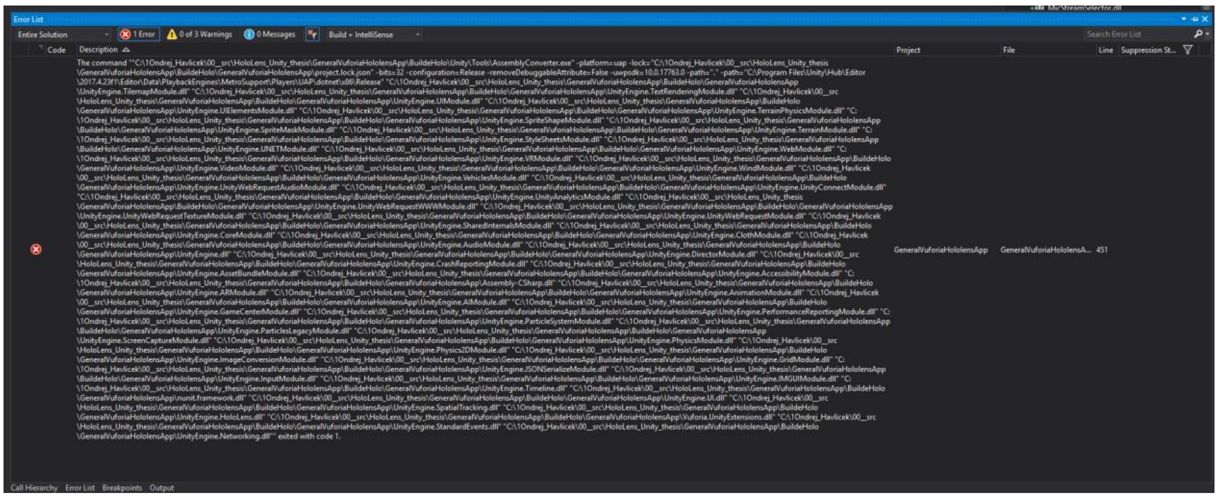


Figure 5.12 Unity play mode with Mixed Reality Camera

5.2.1.2 Errors During Compilation of Vuforia and HoloToolkit in the Same Project

When developing HoloLens application with the HoloToolkit framework everything worked fine, but when the Vuforia plugin needed to be added to the project some components needed to be changed. The recommended camera setup according to HoloToolkit is to use the Mixed Reality Camera Parent prefab. The recommended camera setup according to Vuforia is to use Vuforia's ARCamera prefab. After the code examination it was decided to use the ARCamera prefab because it has the same properties as the Mixed Reality Camera plus Vuforia Behavior

Component attached. Unfortunately, when exporting the project to Visual Studio and then compiling the generated project, an unexpected error occurred. The error is visualized in the following figure.



```
The command ""C:\1Ondrej_Havlicek\00__src\HoloLens_Unity_thesis\GeneralVuforiaHololensApp\BuildeHolo\Unity\Tools\AssemblyConverter.exe" -platform=uap -lock="C:\1Ondrej_Havlicek\00__src\HoloLens_Unity_thesis
\GeneralVuforiaHololensApp\BuildeHolo\GeneralVuforiaHololensApp\project.lock.json" -bits=32 -configuration=Release -removeDebuggableAttribute=False -uwpsdk=10.0.17763.0 -path="." -path="C:\Program Files\Unity\Hub\Editor
\2017.4.23f1\Editor\Data\PlaybackEngines\MetroSupport\Players\UAP\dotnet\x86\Release" "C:\1Ondrej_Havlicek\00__src\HoloLens_Unity_thesis\GeneralVuforiaHololensApp\BuildeHolo\GeneralVuforiaHololensApp
\UnityEngine.TilemapModule.dll" "C:\1Ondrej_Havlicek\00__src\HoloLens_Unity_thesis\GeneralVuforiaHololensApp\BuildeHolo\GeneralVuforiaHololensApp\UnityEngine.TextRenderingModule.dll" "C:\1Ondrej_Havlicek\00__src
\HoloLens_Unity_thesis\GeneralVuforiaHololensApp\BuildeHolo\GeneralVuforiaHololensApp\UnityEngine.UIModule.dll" "C:\1Ondrej_Havlicek\00__src\HoloLens_Unity_thesis\GeneralVuforiaHololensApp\BuildeHolo
\GeneralVuforiaHololensApp\UnityEngine.UIElementsModule.dll" "C:\1Ondrej_Havlicek\00__src\HoloLens_Unity_thesis\GeneralVuforiaHololensApp\BuildeHolo\GeneralVuforiaHololensApp\UnityEngine.TerrainPhysicsModule.dll" "C:
\1Ondrej_Havlicek\00__src\HoloLens_Unity_thesis\GeneralVuforiaHololensApp\BuildeHolo\GeneralVuforiaHololensApp\UnityEngine.SpriteShapeModule.dll" "C:\1Ondrej_Havlicek\00__src\HoloLens_Unity_thesis\GeneralVuforiaHololensApp
\BuildeHolo\GeneralVuforiaHololensApp\UnityEngine.SpriteMaskModule.dll" "C:\1Ondrej_Havlicek\00__src\HoloLens_Unity_thesis\GeneralVuforiaHololensApp\BuildeHolo\GeneralVuforiaHololensApp\UnityEngine.TerrainModule.dll" "C:
\1Ondrej_Havlicek\00__src\HoloLens_Unity_thesis\GeneralVuforiaHololensApp\BuildeHolo\GeneralVuforiaHololensApp\UnityEngine.StyleSheetsModule.dll" "C:\1Ondrej_Havlicek\00__src\HoloLens_Unity_thesis\GeneralVuforiaHololensApp
\BuildeHolo\GeneralVuforiaHololensApp\UnityEngine.UNETModule.dll" "C:\1Ondrej_Havlicek\00__src\HoloLens_Unity_thesis\GeneralVuforiaHololensApp\BuildeHolo\GeneralVuforiaHololensApp\UnityEngine.WebModule.dll" "C:
\1Ondrej_Havlicek\00__src\HoloLens_Unity_thesis\GeneralVuforiaHololensApp\BuildeHolo\GeneralVuforiaHololensApp\UnityEngine.VRModule.dll" "C:\1Ondrej_Havlicek\00__src\HoloLens_Unity_thesis\GeneralVuforiaHololensApp\BuildeHolo
\GeneralVuforiaHololensApp\UnityEngine.VideoModule.dll" "C:\1Ondrej_Havlicek\00__src\HoloLens_Unity_thesis\GeneralVuforiaHololensApp\BuildeHolo\GeneralVuforiaHololensApp\UnityEngine.WindModule.dll" "C:\1Ondrej_Havlicek
\00__src\HoloLens_Unity_thesis\GeneralVuforiaHololensApp\BuildeHolo\GeneralVuforiaHololensApp\UnityEngine.VehiclesModule.dll" "C:\1Ondrej_Havlicek\00__src\HoloLens_Unity_thesis\GeneralVuforiaHololensApp\BuildeHolo
\GeneralVuforiaHololensApp\UnityEngine.UnityWebRequestAudioModule.dll" "C:\1Ondrej_Havlicek\00__src\HoloLens_Unity_thesis\GeneralVuforiaHololensApp\BuildeHolo\GeneralVuforiaHololensApp\UnityEngine.UnityConnectModule.dll"
"C:\1Ondrej_Havlicek\00__src\HoloLens_Unity_thesis\GeneralVuforiaHololensApp\BuildeHolo\GeneralVuforiaHololensApp\UnityEngine.UnityAnalyticsModule.dll" "C:\1Ondrej_Havlicek\00__src\HoloLens_Unity_thesis
\GeneralVuforiaHololensApp\BuildeHolo\GeneralVuforiaHololensApp\UnityEngine.UnityWebRequestWWWModule.dll" "C:\1Ondrej_Havlicek\00__src\HoloLens_Unity_thesis\GeneralVuforiaHololensApp\BuildeHolo\GeneralVuforiaHololensApp
\UnityEngine.UnityWebRequestTextureModule.dll" "C:\1Ondrej_Havlicek\00__src\HoloLens_Unity_thesis\GeneralVuforiaHololensApp\BuildeHolo\GeneralVuforiaHololensApp\UnityEngine.UnityWebRequestModule.dll" "C:\1Ondrej_Havlicek
\00__src\HoloLens_Unity_thesis\GeneralVuforiaHololensApp\BuildeHolo\GeneralVuforiaHololensApp\UnityEngine.SharedInternalsModule.dll" "C:\1Ondrej_Havlicek\00__src\HoloLens_Unity_thesis\GeneralVuforiaHololensApp\BuildeHolo
\GeneralVuforiaHololensApp\UnityEngine.CoreModule.dll" "C:\1Ondrej_Havlicek\00__src\HoloLens_Unity_thesis\GeneralVuforiaHololensApp\BuildeHolo\GeneralVuforiaHololensApp\UnityEngine.ClothModule.dll" "C:\1Ondrej_Havlicek
\00__src\HoloLens_Unity_thesis\GeneralVuforiaHololensApp\BuildeHolo\GeneralVuforiaHololensApp\UnityEngine.AudioModule.dll" "C:\1Ondrej_Havlicek\00__src\HoloLens_Unity_thesis\GeneralVuforiaHololensApp\BuildeHolo
\GeneralVuforiaHololensApp\UnityEngine.dll" "C:\1Ondrej_Havlicek\00__src\HoloLens_Unity_thesis\GeneralVuforiaHololensApp\BuildeHolo\GeneralVuforiaHololensApp\UnityEngine.DirectorModule.dll" "C:\1Ondrej_Havlicek\00__src
\HoloLens_Unity_thesis\GeneralVuforiaHololensApp\BuildeHolo\GeneralVuforiaHololensApp\UnityEngine.CrashReportingModule.dll" "C:\1Ondrej_Havlicek\00__src\HoloLens_Unity_thesis\GeneralVuforiaHololensApp\BuildeHolo
\GeneralVuforiaHololensApp\UnityEngine.AssetBundleModule.dll" "C:\1Ondrej_Havlicek\00__src\HoloLens_Unity_thesis\GeneralVuforiaHololensApp\BuildeHolo\GeneralVuforiaHololensApp\UnityEngine.AccessibilityModule.dll" "C:
\1Ondrej_Havlicek\00__src\HoloLens_Unity_thesis\GeneralVuforiaHololensApp\BuildeHolo\GeneralVuforiaHololensApp\Assembly-CSharp.dll" "C:\1Ondrej_Havlicek\00__src\HoloLens_Unity_thesis\GeneralVuforiaHololensApp\BuildeHolo
\GeneralVuforiaHololensApp\UnityEngine.ARModule.dll" "C:\1Ondrej_Havlicek\00__src\HoloLens_Unity_thesis\GeneralVuforiaHololensApp\BuildeHolo\GeneralVuforiaHololensApp\UnityEngine.AnimationModule.dll" "C:\1Ondrej_Havlicek
\00__src\HoloLens_Unity_thesis\GeneralVuforiaHololensApp\BuildeHolo\GeneralVuforiaHololensApp\UnityEngine.AIModule.dll" "C:\1Ondrej_Havlicek\00__src\HoloLens_Unity_thesis\GeneralVuforiaHololensApp\BuildeHolo
\GeneralVuforiaHololensApp\UnityEngine.GameCenterModule.dll" "C:\1Ondrej_Havlicek\00__src\HoloLens_Unity_thesis\GeneralVuforiaHololensApp\BuildeHolo\GeneralVuforiaHololensApp\UnityEngine.PerformanceReportingModule.dll" "C:
\1Ondrej_Havlicek\00__src\HoloLens_Unity_thesis\GeneralVuforiaHololensApp\BuildeHolo\GeneralVuforiaHololensApp\UnityEngine.ParticleSystemModule.dll" "C:\1Ondrej_Havlicek\00__src\HoloLens_Unity_thesis\GeneralVuforiaHololensApp
\BuildeHolo\GeneralVuforiaHololensApp\UnityEngine.ParticlesLegacyModule.dll" "C:\1Ondrej_Havlicek\00__src\HoloLens_Unity_thesis\GeneralVuforiaHololensApp\BuildeHolo\GeneralVuforiaHololensApp
\UnityEngine.ScreenCaptureModule.dll" "C:\1Ondrej_Havlicek\00__src\HoloLens_Unity_thesis\GeneralVuforiaHololensApp\BuildeHolo\GeneralVuforiaHololensApp\UnityEngine.PhysicsModule.dll" "C:\1Ondrej_Havlicek\00__src
\HoloLens_Unity_thesis\GeneralVuforiaHololensApp\BuildeHolo\GeneralVuforiaHololensApp\UnityEngine.Physics2DModule.dll" "C:\1Ondrej_Havlicek\00__src\HoloLens_Unity_thesis\GeneralVuforiaHololensApp\BuildeHolo
\GeneralVuforiaHololensApp\UnityEngine.ImageConversionModule.dll" "C:\1Ondrej_Havlicek\00__src\HoloLens_Unity_thesis\GeneralVuforiaHololensApp\BuildeHolo\GeneralVuforiaHololensApp\UnityEngine.GridModule.dll" "C:
\1Ondrej_Havlicek\00__src\HoloLens_Unity_thesis\GeneralVuforiaHololensApp\BuildeHolo\GeneralVuforiaHololensApp\UnityEngine.JSONSerializeModule.dll" "C:\1Ondrej_Havlicek\00__src\HoloLens_Unity_thesis\GeneralVuforiaHololensApp
\BuildeHolo\GeneralVuforiaHololensApp\UnityEngine.InputModule.dll" "C:\1Ondrej_Havlicek\00__src\HoloLens_Unity_thesis\GeneralVuforiaHololensApp\BuildeHolo\GeneralVuforiaHololensApp\UnityEngine.IMGUIModule.dll" "C:
\1Ondrej_Havlicek\00__src\HoloLens_Unity_thesis\GeneralVuforiaHololensApp\BuildeHolo\GeneralVuforiaHololensApp\UnityEngine.Timeline.dll" "C:\1Ondrej_Havlicek\00__src\HoloLens_Unity_thesis\GeneralVuforiaHololensApp\BuildeHolo
\GeneralVuforiaHololensApp\nunit.framework.dll" "C:\1Ondrej_Havlicek\00__src\HoloLens_Unity_thesis\GeneralVuforiaHololensApp\BuildeHolo\GeneralVuforiaHololensApp\UnityEngine.UI.dll" "C:\1Ondrej_Havlicek\00__src
\HoloLens_Unity_thesis\GeneralVuforiaHololensApp\BuildeHolo\GeneralVuforiaHololensApp\UnityEngine.SpatialTracking.dll" "C:\1Ondrej_Havlicek\00__src\HoloLens_Unity_thesis\GeneralVuforiaHololensApp\BuildeHolo
\GeneralVuforiaHololensApp\UnityEngine.HoloLens.dll" "C:\1Ondrej_Havlicek\00__src\HoloLens_Unity_thesis\GeneralVuforiaHololensApp\BuildeHolo\GeneralVuforiaHololensApp\Vuforia.UnityExtensions.dll" "C:\1Ondrej_Havlicek\00__src
\HoloLens_Unity_thesis\GeneralVuforiaHololensApp\BuildeHolo\GeneralVuforiaHololensApp\UnityEngine.StandardEvents.dll" "C:\1Ondrej_Havlicek\00__src\HoloLens_Unity_thesis\GeneralVuforiaHololensApp\BuildeHolo
\GeneralVuforiaHololensApp\UnityEngine.Networking.dll"" exited with code 1.
```

GeneralVuforiaHololensApp GeneralVuforiaHololensA... 451

Figure 5.13 List of Errors During the Vuforia and HoloToolkit Compilation

The figure shows that the AssemblyConverter.exe process failed. The root of this problem was not found but according to Joost van Schaik "*Apparently some cruft stays behind preventing Unity from rebuilding the app the second time around. This is because Unity does not always overwrite all the files, presumably to speed up the build process that second time*." [44]. The solution was tested but it didn't work for this case.

The problem was solved if the Vuforia plugin was compiled first, without the HoloToolkit framework inside the project. A general process was developed to make sure that the problems with compilation were avoided.

General Process of Developing Vuforia Application with HoloToolkit Framework

1. Create new 3D project
2. Switch target platform to UWP in Build Settings
3. Player Settings:
   a. Activate Vuforia in your project
      i. Player Settings -> UWP window -> XR settings -> enable Vuforia
   b. Set run in background to true in Resolution and Presentation
   c. Check following capabilities in publish settings
      i. Internet client
      ii. Webcam
4. Export and build the project
5. Import HoloToolkit package
6. Set up project using commands
   a. Mixed Reality toolkit-> Configure -> Apply mixed reality project settings -> uncheck spatial mapping (it is not required for this app)
   b. Mixed Reality toolkit-> Configure -> Apply mixed reality scene settings -> no spatial prefab (no need)

7. Save Scene to asset folder
8. Delete MixedRealityCameraParent
9. Add Vuforia Camera: GameObject->Vuforia->AR Camera
    a. This will import some scripts and assets
10. Configure Vuforia
    a. Window->Vuforia Configuration->Digital Eyewear
        i. Type: Digital eyewear
        ii. Device config: HoloLens
11. Set up extended tracking
    a. Window->Vuforia Configuration->Device tracker
        i. Track pose: true
        ii. Tracking mode: Positional
        iii. Set up extended tracking to each image target
12. Add image target to Scene
    a. GameObject->Vuforia->Image
13. Add asset to image target child to see after image found
14. In case you want custom image target, do not forget to load it and allow it in Vuforia configuration
15. Build Application normal

Using this process, the application is compiled without errors.

5.2.1.3 Communication Using C# Sockets in Unity

The Unity API doesn't provide an easy way to set up a socket to connect to specified IP and port number because the unity tries to hide these low-level details from the Unity developer.

Originally it was intended to use .NET Standard C# library *System.Networking.Sockets* but this library is not supported in Unity and therefore a platform specific implementation needs to be used. UWP application has their own implementation of sockets and they doesn't support the standart *System.Networking.Sockets* library as well. The UWP applications use *Windows.Networking.Sockets* library which supports C# asynchronous keywords `await` and `asynch`. These keywords were introduced in C# 5 which is not supported in Unity 2017.4. Therefore, the keywords cannot be used in the Unity editor, but they can be used in the generated application for UWP application. This is where the preprocessor directives come to action.

In order to use Sockets in HoloLens application a preprocessor directive `NETFX_CORE` had to be used. This directive ensures that the code, enclosed by this directive in if statement, is compiled for UWP applications when using .NET scripting backend. A `StreamSocket` class from the standard UWP library is used for communication.

5.2.1.4 Application Failure When the Connection is Not Established

When the user requests communication with the server and the server is not available, the connection request is canceled after 7 seconds. Although the application correctly writes an output to the console that the time out was detected, the application then throws

`NullReferenceException` and terminates itself. This error was not fixed, and the application will terminate itself if the connection is not established.

The class `SimpleStreamSocket` was tested on a PC computer in a different UWP application so the problem is probably not in the class itself. The problem might be in using `asynch` and `await` keywords and canceling a .NET standard `Task` class during execution in Unity.

5.2.1.5 Firewall Blocking the Communication

When the HoloLens application tried to connect to the server application on notebook via Wi-Fi, a firewall blocked any attempts to establish the communication.

This problem was solved by adding an inbound rule in a Windows application called Windows Defender Firewall with Advanced Security on the notebook. This application oversees the firewall settings on the target notebook.

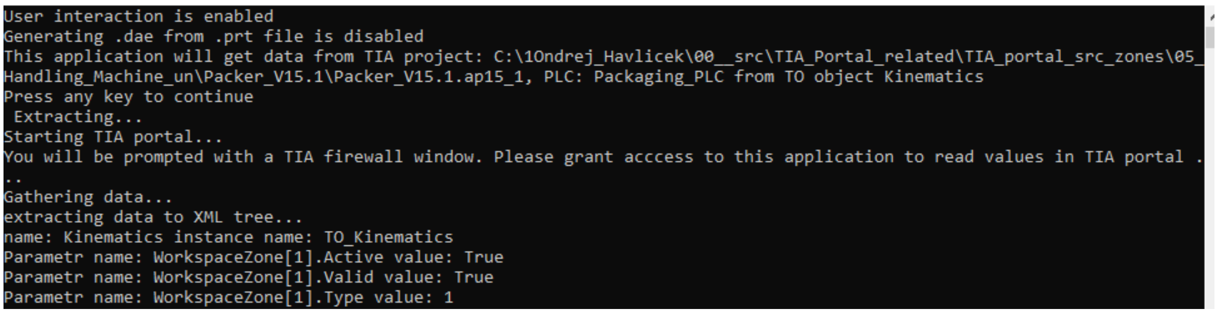
The rule was to allow access for an executable file "SynchronousSocketServer.exe" to accept any incoming TCP messages at a specified port on private networks. When the firewall was set up, the communication worked smoothly.

6 Application Demonstration

This chapter demonstrates the application behavior.

6.1 Server Application

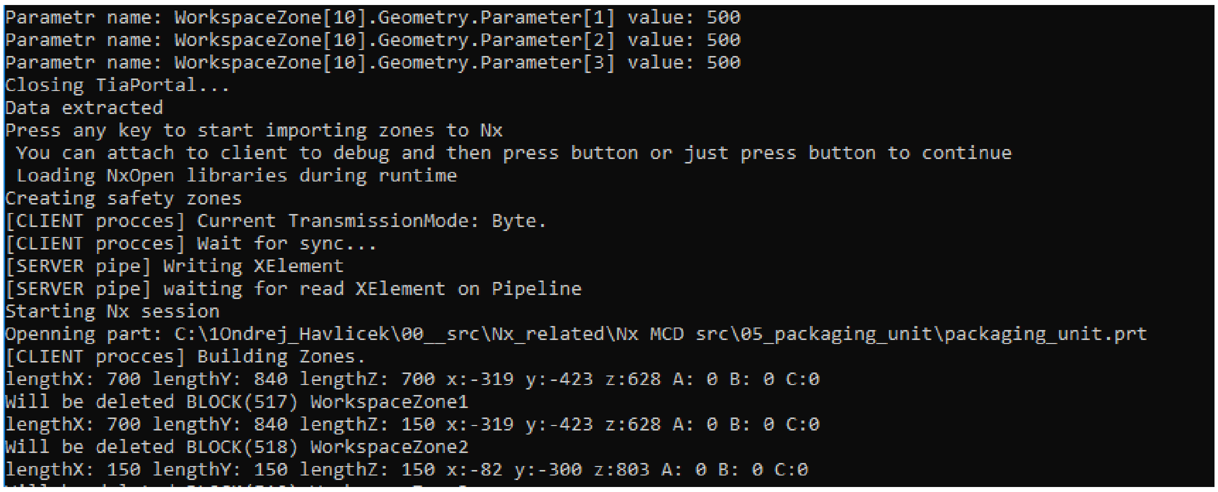
The first figure shows the output during the data extraction from the TIA Portal.



```
User interaction is enabled
Generating .dae from .prt file is disabled
This application will get data from TIA project: C:\1Ondrej_Havlicek\00__src\TIA_Portal_related\TIA_portal_src_zones\05_
Handling_Machine_un\Packer_V15.1\Packer_V15.1.ap15_1, PLC: Packaging_PLC from TO object Kinematics
Press any key to continue
 Extracting...
Starting TIA portal...
You will be prompted with a TIA firewall window. Please grant acccess to this application to read values in TIA portal .
..
Gathering data...
extracting data to XML tree...
name: Kinematics instance name: TO_Kinematics
Parametr name: WorkspaceZone[1].Active value: True
Parametr name: WorkspaceZone[1].Valid value: True
Parametr name: WorkspaceZone[1].Type value: 1
```

Figure 6.1 Server application output part 1

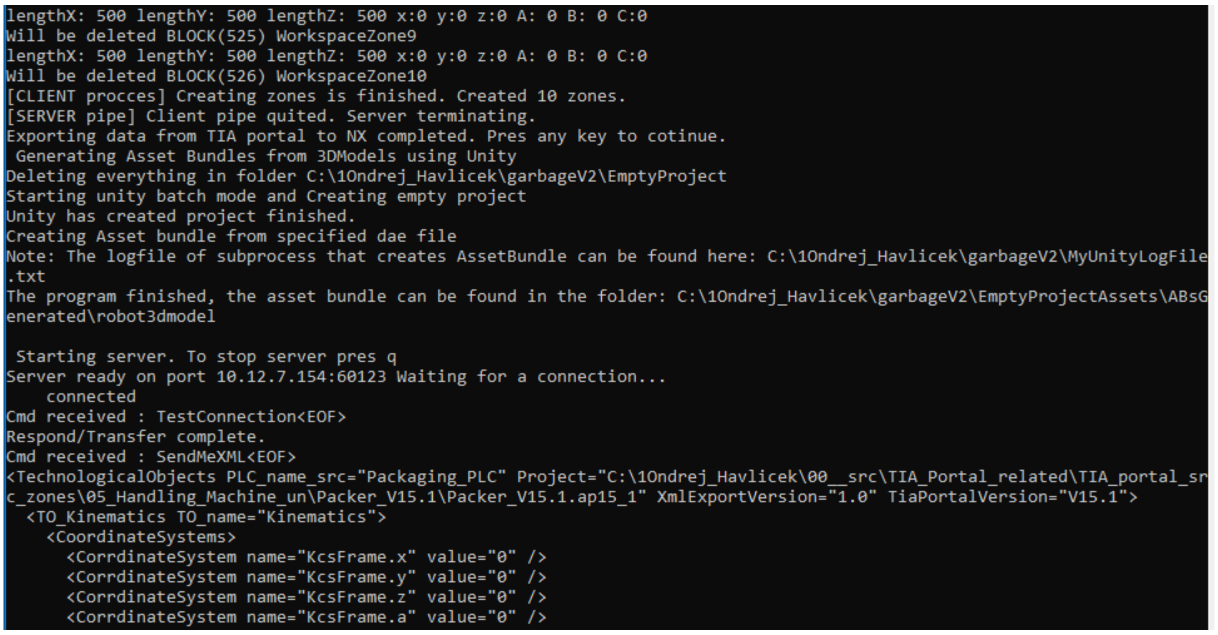
The second figure shows the output when the TIA Portal extraction is done which is followed by the import of work space data into NX. The user can see the communication via Anonymous pipes between the two processes. This communication is followed by old work space zones being deleted and the creation of new ones.



```
Parametr name: WorkspaceZone[10].Geometry.Parameter[1] value: 500
Parametr name: WorkspaceZone[10].Geometry.Parameter[2] value: 500
Parametr name: WorkspaceZone[10].Geometry.Parameter[3] value: 500
Closing TiaPortal...
Data extracted
Press any key to start importing zones to Nx
 You can attach to client to debug and then press button or just press button to continue
 Loading NxOpen libraries during runtime
Creating safety zones
[CLIENT procces] Current TransmissionMode: Byte.
[CLIENT procces] Wait for sync...
[SERVER pipe] Writing XElement
[SERVER pipe] waiting for read XElement on Pipeline
Starting Nx session
Openning part: C:\1Ondrej_Havlicek\00__src\Nx_related\Nx MCD src\05_packaging_unit\packaging_unit.prt
[CLIENT procces] Building Zones.
lengthX: 700 lengthY: 840 lengthZ: 700 x:-319 y:-423 z:628 A: 0 B: 0 C:0
Will be deleted BLOCK(517) WorkspaceZone1
lengthX: 700 lengthY: 840 lengthZ: 150 x:-319 y:-423 z:628 A: 0 B: 0 C:0
Will be deleted BLOCK(518) WorkspaceZone2
lengthX: 150 lengthY: 150 lengthZ: 150 x:-82 y:-300 z:803 A: 0 B: 0 C:0
```

Figure 6.2 Server application output part 2

The third figure shows the result of the zone creation in NX and the communication between processes. After the zones are created, a process for creating an Asset Bundle starts. When the Asset Bundle is created, the application starts to act as a server and starts to listen at the specified IP and port. The figure demonstrates partial output of what happens when the HoloLens applications connects.



```
lengthX: 500 lengthY: 500 lengthZ: 500 x:0 y:0 z:0 A: 0 B: 0 C:0
Will be deleted BLOCK(525) WorkspaceZone9
lengthX: 500 lengthY: 500 lengthZ: 500 x:0 y:0 z:0 A: 0 B: 0 C:0
Will be deleted BLOCK(526) WorkspaceZone10
[CLIENT procces] Creating zones is finished. Created 10 zones.
[SERVER pipe] Client pipe quited. Server terminating.
Exporting data from TIA portal to NX completed. Pres any key to cotinue.
 Generating Asset Bundles from 3DModels using Unity
Deleting everything in folder C:\1Ondrej_Havlicek\garbageV2\EmptyProject
Starting unity batch mode and Creating empty project
Unity has created project finished.
Creating Asset bundle from specified dae file
Note: The logfile of subprocess that creates AssetBundle can be found here: C:\1Ondrej_Havlicek\garbageV2\MyUnityLogFile
.txt
The program finished, the asset bundle can be found in the folder: C:\1Ondrej_Havlicek\garbageV2\EmptyProjectAssets\ABsG
enerated\robot3dmodel

 Starting server. To stop server pres q
Server ready on port 10.12.7.154:60123 Waiting for a connection...
    connected
Cmd received : TestConnection<EOF>
Respond/Transfer complete.
Cmd received : SendMeXML<EOF>
<TechnologicalObjects PLC_name_src="Packaging_PLC" Project="C:\1Ondrej_Havlicek\00__src\TIA_Portal_related\TIA_portal_sr
c_zones\05_Handling_Machine_un\Packer_V15.1\Packer_V15.1.ap15_1" XmlExportVersion="1.0" TiaPortalVersion="V15.1">
  <TO_Kinematics TO_name="Kinematics">
    <CoordinateSystems>
      <CorrdinateSystem name="KcsFrame.x" value="0" />
      <CorrdinateSystem name="KcsFrame.y" value="0" />
      <CorrdinateSystem name="KcsFrame.z" value="0" />
      <CorrdinateSystem name="KcsFrame.a" value="0" />
```

Figure 6.3 Server application output part 3

6.2 HoloLens Application

When the application starts, the user see a “Look for Target Image” message as is shown in Figure 6.4. This tells the user to look for the QR code.

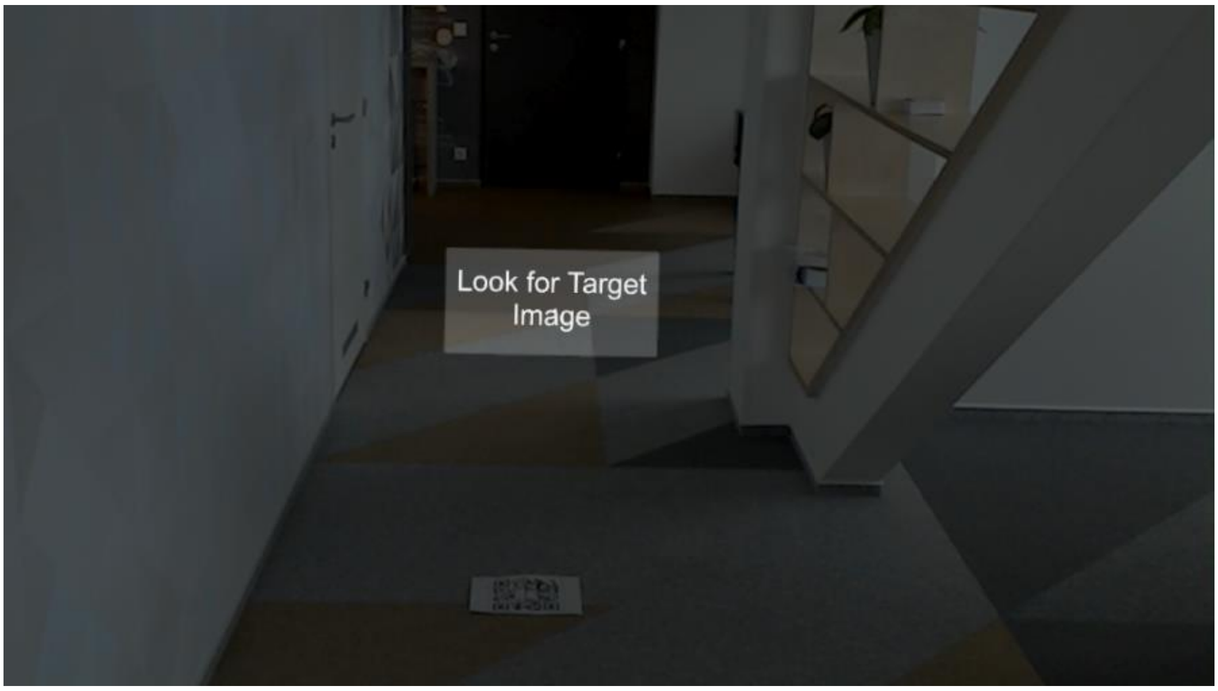


Figure 6.4 The “Look for Target Image” message

In order to visualize the application, a printed QR code needs to be placed on the floor. This QR code serves as a point of reference.

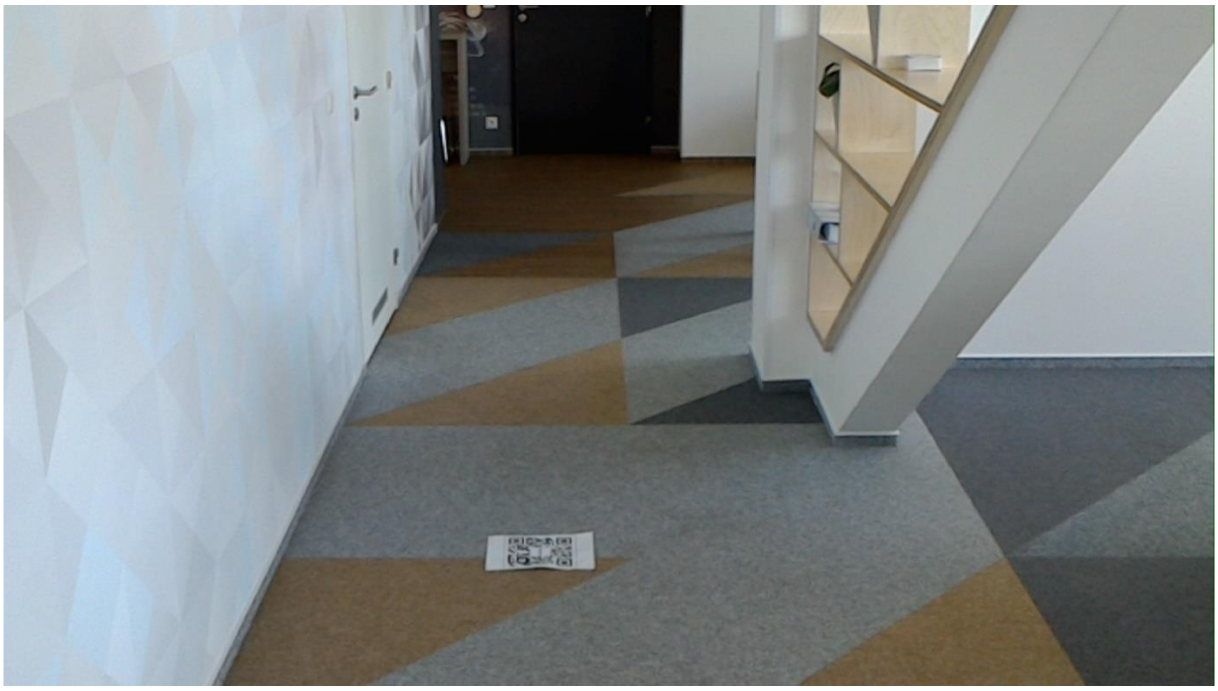


Figure 6.5 Photo of the real space

The QR code is visualized in the following figure.

Figure 6.6 QR code used for reference point

In the next figure is a photo taken during the application runtime. In the picture there is a part of an assembly line and three work space zones. Every zone has its menu bar which controls the zone. It is possible to see that the application gives a good idea of how the zones are related to the assembly line and to the real space, which fulfills one of the use cases defined in section 2.1.

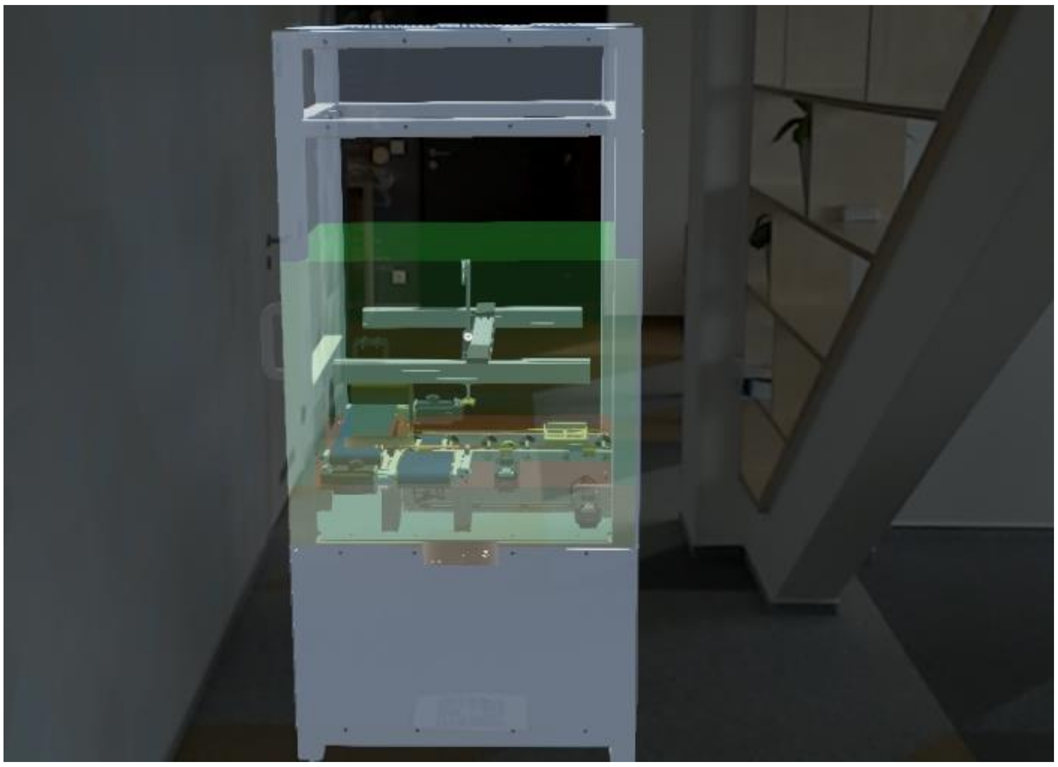


Figure 6.7 HoloLens application with 3D model and imported work space zones

7 Conclusion

This thesis aimed to integrate augmented reality into the digital workflow of virtual commissioning. The application described in the chapter 2.2 was successfully developed and implemented. Therefore, the thesis fulfilled the defined target. The created application satisfy the defined use case and it is fully integrated in workflow of virtual commissioning.

Based on the feedback given by four of the Siemens's employees it can be concluded that the application works as described and it is useful for adjusting workspace zones and synchronizing the data. But thorough research needs to be done among the end-users to verify this claim.

Based on our experience with employees, the biggest issue with this application is that it is a new technology and not many people are used to working with it. It takes time for the user to get used to working with the basic tap and tap & hold commands. This might demotivate the users to utilize the developed HoloLens application.

7.1 Future Work

It is planned to improve the communication and then test the application in real environment to test the user experience and get the necessary feedback from the end-users.

For the field application it is planned to use OPC UA communication standard for communication since it is becoming one of the main standards for communication [45]. The current implementation does not use a secure connection and therefore it is vulnerable to any potential attacks.

The application will also need to resolve the problem with the Gimbal Lock phenomenon in order to be able to change the orientation in HoloLens application and then update these changes back to other software solutions.

The development of HoloToolkit framework has been terminated and therefore, the application will be upgraded to Mixed Reality Toolkit (MRTK) when the first stable release of MRTK is released.

The last improvement is related to coordinate systems defined in TIA Portal. Implementing these coordinate systems would simplify the zones configuration. The coordinate systems are listed in the section 3.6 in the Table 3-2. The data about these coordinate systems is already included in the XML export.

8 Bibliography

1. *Virtual Commissioning of an Assembly Cell with.* **Makris, S., Michalos, G. and Chryssolouri, and G.** s.l. : Hindawi Publishing Corporation, 2012, Advances in Decision Sciences, Vol. 2012. ID 428060.

2. *Virtual Commissioning Of Manufacturing Systems A Review And New Approaches For Simplification.* **Hoffmann, Peter, et al.** Kuala Lumpur, Malaysia : European Council for Modelling, 2010. pp. 175-181. ISBN: 978-0-9564944-0-5.

3. *A comparison between three SDLC models waterfall model, spiral model, and Incremental/Iterative model.* **Alshamrani, Adel and Bahattab, Abdullah.** 1, s.l. : International Journal of Computer Science Issues (IJCSI), January 2015, Vol. 12, pp. 106-111. ISSN 1694-0814.

4. *Impact of Requirements Elicitation Processes on Success of Information System Development Projects.* **Bormane, Līga, et al.** 1, Riga : Information Technology and Management Science, December 2016, Information Technology and Management Science, Vol. 19, pp. 57-64. ISSN 2255-9094.

5. **Schloegl, Dr. Wolfgang.** *Automation Designer integrating Electrics and Automation in Digital Manufacturing.* Siemens AG. 2018. Tech. rep.

6. **EPLAN.** EPLAN - efficient engineering. [Online] EPLAN . https://www.eplanusa.com.

7. *Industry use of virtual reality in product design and manufacturing: a survey.* **Berg, Leif P. and Vance, Judy M.** 3 01, 2017, Virtual Reality, Vol. 21, pp. 1-17. DOI:10.1007/s10055-016-0293-9.

8. **HTC Corporation.** Vive. *Vive Product comparsion.* [Online] HTC Corporation, 2019. https://www.vive.com/us/comparison/.

9. **Facebook Technologies, LLC.** Oculus Homepage. *Oculus.* [Online] Facebook Technologies, LLC., 2019. https://www.oculus.com/.

10. **Samsung Electronics.** Samsung Gear VR. *Samsung.* [Online] Samsung Electronics, 2019. https://www.samsung.com/global/galaxy/gear-vr/.

11. **Kore, Akshay.** Understanding the different types of AR devices. *UX Design.* [Online] UX Collective, September 9, 2018. [Cited: May 9, 2019.] https://uxdesign.cc/augmented-reality-device-types-a7668b15bf7a.

12. *Review on Application of Augmented Reality in Civil Engineering.* **Agarwal, Siddhant.** 2016. International Conference on Inter Disciplinary Research in Engineering and Technology. pp. 68-71. ISBN: 978-81-929866-0-9.

13. *Review and analysis of augmented reality literature for construction industry.* **Rankohi, Sara and Waugh, Lloyd.** s.l. : Visualization in Engineering, 8 29, 2013, Visualization in Engineering, Vol. 1, p. 9. ISSN: 2213-7459.

14. **Leswing, Kif.** 'Pokémon Go' was the most downloaded iPhone app worldwide in 2016, Apple says. *Business Insider.* [Online] Business Insider Deutschland, 1 5, 2017. https://www.businessinsider.de/pokemon-go-most-downloaded-ios-app-worldwide-2016-2017-1?r=US&IR=T.

15. **Delgado, Rick.** 8 Real-World Uses for Microsoft HoloLens. *Makeuseof.com.* [Online] Makeuseof, 2 23, 2015. https://www.makeuseof.com/tag/8-real-world-uses-microsoft-hololens/.

16. **Terry, Ruth.** IKEA's New AR App Might Save You a Trip to the Store. *Interesting engineering.* [Online] Interesting engineering, 10 20, 2017. https://interestingengineering.com/ikeas-new-ar-app-might-save-you-a-trip-to-the-store.

17. **Takahashi, Dean.** John Riccitiello sets out to identify the engine of growth for Unity Technologies (interview). *VentureBeat.com.* [Online] VentureBeat, October 23, 2014. [Cited: May 3, 2019.] https://venturebeat.com/2014/10/23/john-riccitiello-sets-out-to-identify-the-engine-of-growth-for-unity-technologies-interview/.

18. **Unity Technologies.** Unity. *Unity3d.com.* [Online] 2019. [Cited: May 4, 2019.] https://unity3d.com/unity.

19. —. Unity User Manual (2017.4). *Unity Documentation.* [Online] Unity Technologies, 2017. https://docs.unity3d.com/2017.4/Documentation/Manual/index.html.

20. —. Unity Scripting Reference. *Unity Documentation.* [Online] 2017. [Cited: May 2, 2019.] https://docs.unity3d.com/2017.4/Documentation/ScriptReference/.

21. **PTC Inc.** Vuforia Fusion. *Vuforia Developer Library.* [Online] PTC Inc., 2018. [Cited: May 5, 2019.] https://library.vuforia.com/content/vuforia-library/en/articles/Training/vuforia-fusion-article.html.

22. *Simultaneous localization and mapping: part I.* **Durrant-Whyte, H. and Bailey, T.** 6 2006, IEEE Robotics Automation Magazine, Vol. 13, pp. 99-110. ISSN: 1070-9932.

23. *Robust Real-Time Visual Odometry with a Single Camera and an IMU.* **Kneip, Laurent, Chli, Margarita and Siegwart, Roland.** 2011.

24. **Rubino, Daniel.** https://www.windowscentral.com/microsoft-hololens-processor-storage-and-ram. *Windowscentral.* [Online] Windows Central, May 2, 2016. [Cited: May 6, 2019.] https://www.windowscentral.com/microsoft-hololens-processor-storage-and-ram.

25. **Microsoft Corporation.** Mixed Reality documentation. *Mircrosoft Documentation.* [Online] Microsoft, March 21, 2018. [Cited: May 6, 2019.] https://docs.microsoft.com/en-us/windows/mixed-reality/.

26. **Stein, Scott.** Microsoft HoloLens: Not holograms, exactly, but strike one in AR turf war. *CNET.* [Online] CBS Interactive Inc., January 21, 2015. [Cited: May 7, 2019.] https://www.cnet.com/news/microsoft-hololens-not-a-hologram-exactly-but-another-entry-in-an-augmented-reality-turf-war/.

27. **Merriam-Webster.** Hologram | Definition of Hologram by Merriam-Webster. *Dictionary by Merriam-Webster: America's most-trusted online dictionary.* [Online] Merriam-Webster, Inc. [Cited: May 9, 2019.] https://www.merriam-webster.com/dictionary/hologram.

28. **Microsoft Corporation.** MixedRealityToolkit-Unity. *Github.* [Online] December 5, 2018. [Cited: April 25, 2019.] https://github.com/microsoft/MixedRealityToolkit-Unity/tree/htk_release.

29. —. Universal Windows Platform documentation. *Microsoft Docs.* [Online] Microsoft Corporation. [Cited: May 8, 2019.] https://docs.microsoft.com/en-us/windows/uwp/.

30. —. .NET API Browser. *Microsoft Docs.* [Online] Microsoft Corporation. [Cited: May 8, 2019.] https://docs.microsoft.com/en-us/dotnet/api/?view=netframework-4.8.

31. —. Pipes. *Microsoft Docs.* [Online] Microsoft Corporation, May 31, 2018. [Cited: May 8, 2019.] https://docs.microsoft.com/en-us/windows/desktop/ipc/pipes.

32. **Siemens AG.** S7-1500T Kinematics Functions V4.0. *Siemens Industry Online Support.* [Online] November 27, 2017. [Cited: May 8, 2019.] https://support.industry.siemens.com/cs/attachments/109749264/s7-1500t_kinematic_function_manual_en-US_en-US.pdf?download=true. A5E42062707-AA.

33. —. TIA Portal Openness: Introduction and Demo Application. *Siemens Industry Online Support.* [Online] Siemens AG, March 5, 2019. [Cited: May 2, 2019.]

https://support.industry.siemens.com/cs/document/108716692/tia-portal-openness%3A-introduction-and-demo-application. 108716692.

34. **Company, Mecademic.** How is orientation in space represented with Euler angles? *Mecademic Indrustrial Robot Arms.* [Online] Mecademic, 5 2, 2019. [Cited: May 9, 2019.] https://www.mecademic.com/resources/Euler-angles/Euler-angles.

35. **Meisen, Wolfgang.** Siemens and KUKA announce cooperation. [Online] Siemens Media Relations, 9 17, 2013. https://www.siemens.com/press/pi/IDT2013094095e.

36. **Support, Siemens Industry Online.** Controlling a KUKA Industrial Robot Using a SIMATIC S7-1500. 8 10, 2018.

37. **Jia, Yan-Bin.** Quaternions. *Problem Solving Techniques for Applied Computer Science.* [Online] September 4, 2018. [Cited: May 2, 2019.] http://web.cs.iastate.edu/~cs577/handouts/quaternion.pdf.

38. **Gregory, Douglas.** Converting a quaternion in a right to left handed coordinate system. *Game Development Stack Exchange.* [Online] Stack Overflow, April 28, 2018. [Cited: May 8, 2019.] https://gamedev.stackexchange.com/a/157954.

39. **Blanco, José-Luis.** *A tutorial on SE(3) transformation parameterizations and on-manifold optimization.* University of Malaga. 2010. Tech. rep.

40. **Siemens AG.** SIMATIC Openness: Automating creation of projects. *Siemens Industry Online Support.* [Online] September 9, 2018. [Cited: May 8, 2019.] https://support.industry.siemens.com/cs/document/109477163/simatic-openness%3A-automating-creation-of-projects?dti=0&lc=en-US. 109477163.

41. **Siemens Product Lifecycle Management Software Inc.** NX Open for .NET - Executing Batch Applications. *Documentation Center.* [Online] 2017. [Cited: May 9, 2019.] https://docs.plm.automation.siemens.com/tdoc/nx/12/nx_api#uid:xid1162445:index_nxopen_prog_guide:id1142076:id1253956:genid_for_net_22_1442 .

42. **Metaverse Technologies FRANCE.** PiXYZ Software. *PiXYZ Software.* [Online] Metaverse Technologies FRANCE, 2019. [Cited: May 9, 2019.] https://www.pixyz-software.com/.

43. **ISO.** Industrial automation systems and integration — COLLADA digital asset schema specification for 3D visualization of industrial data. *International Organization for*

Standardization. [Online] 07 July, 2012. [Cited: May 9, 2019.] https://www.iso.org/standard/59902.html. ISO/PAS 17506:2012.

44. **Schaik, Joost van.** Fixing the "...Unity\Tools\AssemblyConverter.exe exited with code 1" Error in Mixed Reality Apps. *DZone.* [Online] May 24, 2018. [Cited: May 5, 2019.] https://dzone.com/articles/fixing-the-quotunitytoolsassemblyconverterexe-exit.

45. **OPC Foundation.** Major Automation Industry Players join OPC UA including TSN initiative. *OPC Foundation Website.* [Online] OPC Foundation, November 27, 2018. [Cited: May 10, 2019.] https://opcfoundation.org/news/press-releases/major-automation-industry-players-join-opc-ua-including-tsn-initiative/.

46. *Simultaneous orthogonal rotation angle.* **Tomažič, Sašo and Stančin, Sara.** Slovenia : The Elektrotehniška zveza Slovenije, 2011, Elektrotehniski Vestnik/Electrotechnical Review, Vol. 78, pp. 7-11. 0013-5852.

47. **Spence, Ewan.** Microsoft HoloLens Review: Winning The Reality Wars. [Online] 1 14, 2017. https://www.forbes.com/sites/ewanspence/2017/01/14/microsoft-hololens-review-experience-review/.

48. **Fitzsimmons, Michelle.** Hands on: Microsoft HoloLens review. [Online] 6 17, 2017. https://www.techradar.com/reviews/wearables/microsoft-hololens-1281834/review.

49. **DeTellem, John.** *Automation Simulation: Your Gateway into Smart Manufacturing.* Siemens Industry, Inc. 2017. Tech. rep.

50. **KISSEL, Richard L., et al.** *Security considerations in the system development life cycle.* [Document] Gaithersburg : National Institute of Standards of Technology, 2008. MD 20899-8930.

9 List of Figures

10 List of Tables

11 List of Snippets

12 List of Abbreviations

Abbreviation	Term
PC	Personal Computer
TCP	Transmission Control Protocol
HMI	Human Machine Interface
API	Application Programming Interface
HUD	Heads Up Display
TIA	Totally Integrated Automation
MCD	Mechatronics Concept Designer
PLC	Programable Logic Controller
CAD	Computer Aided Design
CAM	Computer Aided Manufacturing
CAE	Computer Aided Engineering
GUI	Graphical User Interface
SDLC	Systems Development Life Cycle
SIL	Software-In-the-Loop
CAVE	Cave Automatic Virtual Environment
AutomationML	Automation Markup Language
MRTK	Mixed Reality ToolKit
SDLC	Software development life cycle

13 Appendix

```
<?xml version="1.0" encoding="utf-8"?>
<TechnologicalObjects PLC_name_src="Packaging_PLC"
Project="C:\1Ondrej_Havlicek\00__src\TIA_Portal_related\TIA_portal_src_zon
es\05_Handling_Machine_un\Packer_V15.1\Packer_V15.1.ap15_1"
XmlExportVersion="1.0" TiaPortalVersion="V15.1">
  <TO_Kinematics TO_name="Kinematics">
    <CoordinateSystems>
<!-- Coordinate systems, not implemented -->
      <CorrdinateSystem name="KcsFrame.x" value="0" />
      <CorrdinateSystem name="KcsFrame.y" value="0" />
      ...
    </CoordinateSystems>
    <Zone zoneID="WorkspaceZone1">
      <TOParam name="WorkspaceZone[1].Active" value="false" />
      <TOParam name="WorkspaceZone[1].Valid" value="false" />
      <TOParam name="WorkspaceZone[1].Type" value="1" />
      <TOParam name="WorkspaceZone[1].ReferenceSystem" value="0" />
      <TOParam name="WorkspaceZone[1].Frame.x" value="0" />
      <TOParam name="WorkspaceZone[1].Frame.y" value="0" />
      <TOParam name="WorkspaceZone[1].Frame.z" value="0" />
      <TOParam name="WorkspaceZone[1].Frame.a" value="0" />
      <TOParam name="WorkspaceZone[1].Frame.b" value="0" />
      <TOParam name="WorkspaceZone[1].Frame.c" value="0" />
      <TOParam name="WorkspaceZone[1].Geometry.Type" value="0" />
      <TOParam name="WorkspaceZone[1].Geometry.Parameter[1]" value="50" />
      <TOParam name="WorkspaceZone[1].Geometry.Parameter[2]" value="50" />
      <TOParam name="WorkspaceZone[1].Geometry.Parameter[3]" value="50" />
    </Zone>
    <Zone zoneID="WorkspaceZone2">
      <TOParam name="WorkspaceZone[2].Active" value="true" />
        ...
    </Zone>
    <Zone zoneID="WorkspaceZone3">
    </Zone>
        ...
    <Zone zoneID="WorkspaceZone10">
    </Zone>
  </TO_Kinematics>
</TechnologicalObjects>
```

Figure 13.1 Collapsed XML format for work space zones data extracted from TIA Portal