POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE E DELL'INFORMAZIONE

Laurea Magistrale In Ingegneria Meccanica

Dynamic representation of factory objects in a virtual reality environment

Supervisor: Marcello Urgo, Walter Terkaj

Semen Polonskii

ID: 10651528

Mingyue Yu

ID:10649931

Academic Year 20/21

ABSTRACT

Due to some weaknesses in virtual factory software tools that nowadays are presented in the market, such as low compatibility, high cost, poor graphical representation, and complexity, the project Virtual Learning Factory Toolkits (VLFT) gave out a solution to solve the problems stated above. The solution is to decouple one software tool into three stages: simulation, data processing, and 3D visualization stages.

The thesis aim is to contribute to the VLFT project which purpose is to support engineering education in manufacturing. The main contribution is done in the part of 3D visualization (increasing animation functionality) as well as the data processing stage (data preparation for the visualization stage). During the development stage tools such as JavaScript, BabylonsJS were used as the main tool whereas Java Modelling Tool (JMT) and Python were used as a validation tool. During the validation, part functionality has been checked. Also, some limitations in the model have been faced primarily related to simulation and visualization stages. Possible solutions were proposed in order to overcome limitations. Also, the scope of the future work has been proposed, which includes further development in the simulation stage, data processing stage as well as visualization stage.

Keywords: Digital Twin, Virtual Factory, Virtual Reality, VLFT, Discrete Event Simulation, 3D Visualisation, Data Transformation, JavaScript, WebGL, BabylonJs, JMT, Python

ACKNOWLEDGMENTS

For writing this thesis paper, we would like to express our appreciation to sincerely all of the people who help us a lot.

First of all, we are extremely grateful to our supervisors for this paper: Professor Marcello Urgo and Professor Walter Terkaj. We appreciate the thesis you offer us which is quite interesting, and we are honored to make some contributions to the project VLFT. Also, you played key roles in giving us a lot of important suggestions and guidance during developing the project.

We're deeply indebted to the developers from project VLFT. Thank you for developing this interesting idea which can let us contribute something to the project and also thank you for the notation and functions that you have done in the VE.bjs which makes a huge convenience for us to refer to when working on it.

We would also like to extend our sincere thanks to our parents. Without your supports, we couldn't be here and study in Politecnico di Milano carefreely.

Also, a special thanks to Politecnico di Milano for giving us a gorgeous academic platform and environment to improve ourselves and lightening our future careers.

Last, we want to thanks our friends Huimin, Alexander, Diego, Kia, and Omar. Thank you for sharing a good time with us and helping us for two years of study and especially in this harsh Covid period. Thank you for being our colleagues and friends.

Semen Polonskii; Mingyue Yu

Milano, April 8th,2021

CONTENT

Abstract 2

Acknowledgements 3

Content 4

Table of Figures 7

List of Tables 9

Chapter 1. Introduction 10

1.1 Background 10

1.2 Thesis Aims 14

1.3 Structure of the Thesis Report 14

Chapter 2. State of Art 15

2.1 Virtual Factory 15

2.1.1 Virtual Factory Modelling 15

2.1.2 Virtual Factory Visualization 16

2.2 Software Tools of Discrete Event Simulation 17

2.2.1 Virtual Factory Software Tools Challenges 17

2.2.2 Comparison of discrete event modelling simulator in the market 18

2.3 Virtual reality in Manufacturing 20

2.3.1 Virtual Reality Gaming Engine 20

2.3.2 WebGL 21

2.3.3 3D WebGL Engine 22

Chapter 3. Problem Statement 23

3.1 Problem Statement 23

3.2 Requirements, Goals and Boundaries 23

3.2.1 Requirements 24

3.2.2 Goals 24

3.2.3 Boundaries 25

3.3 Possible Users 25
Chapter 4. Method 27
4.1 General Approach 27
4.2 Discrete Event Simulator Selection 29
4.3 Data Processing Tool Selection 30
4.4 3D Visualization Software Tool Selection 30
4.5 Introduction of New Approach 31
Chapter 5. Solution 33
5.1 Assumptions 33
5.2 Discrete Event Simulation Stage 33
5.3 Data Processing Stage 34
5.3.1 Action and constants definition 35
5.3.2 Data loading and sorting 37
5.3.3 Mapping entities 37
5.3.4 Data preparation 38
5.3.5 Creation structure of animation events 38
5.3.6 Data serialization 40
5.4 3D Visualization Stage 41
5.4.1 Data Loading and Parsing 41
5.4.1.1 Related .txt File Loading and Parsing 44
5.4.1.2 Timestamp Construction 45
5.4.2 Animation Functionalities 46
5.4.2.1 Developments of the Graphic User Interface of Animation Functionalities 46
5.4.2.2 Animation 50
5.4.2.3 Animation Additive 52
5.4.2.4 Show and Hide 53
5.4.2.5 State Event 55
5.4.2.6 Trails 59

5.4.3 Mesh Picking Functionalities in Graphical User Interface 60
Chapter 6. Experiments and Validation 62
6.1 Main Functionality Check 62
6.2 Limitations 64
Chapter 7. Conclusions 66
7.1 Future Work 66
Reference 67

TABLE OF FIGURES

FIGURE 1 THE EVOLUTION OF MANUFACTURING SYSTEM[2] 11
FIGURE 2 DATA FLOW IN (A) DIGITAL MODEL, (B) DIGITAL SHADOW, (C) DIGITAL TWIN 12
FIGURE 3 DIGITAL TWIN BASED VIRTUAL FACTORY CONCEPT[12] 12
FIGURE 4 SCOPE OF THE APPROACH 27
FIGURE 5 JAVA MODELLING TOOLS 29
FIGURE 6 JMODEL[53] 30
FIGURE 7 THE BABYLON.JS PLAYGROUND[56] 31
FIGURE 8 THE FULL SCOPE OF THE NEW APPROACH 31
FIGURE 9 THE SCOPE OF THE NEW APPLICATION 32
FIGURE 10 MODEL HAS BEEN BUILT IN JMT. 34
FIGURE 11 FLOWLINE OF DATA PROCESSING 35
FIGURE 12 THE "MAIN()" FUNCTION WHICH IS THE MAIN STRUCTURE OF THE SCRIPT 35
FIGURE 13 PRE-DEFINED EVENTS AND ADDITIONAL INFORMATION FOR PALLETS 37
FIGURE 14 PRE-DEFINED EVENTS AND ADDITIONAL INFORMATION FOR WORKING STATIONS 37
FIGURE 15 LOADING AND SORTING 37
FIGURE 16 LOGIC OF MAPPING THE JOB ID 38
FIGURE 17 CODE FOR MAPPING ENTITIES 38
FIGURE 18 CODES FOR THE DATA PREPARATION STAGE 38
FIGURE 19 CODES FOR CHECKING OVERLAPPING 39
FIGURE 20 CODE FOR GENERAL FUNCTION 39
FIGURE 21 CODE FOR SPECIFIC FOR STATIC ENTITIES 39
FIGURE 22 CODE FOR DYNAMIC EVENTS 40
FIGURE 23 CODE FOR SERIALIZATION 41
FIGURE 24 IMPORTING ANIMATION.JSON FILE 41
FIGURE 25 STRUCTURE OF ANIMATION JSON FILE 42
FIGURE 26 CODES FOR LOAD SELECTED ANIMATION.JSON FILE 42
FIGURE 27 LOGIC OF SAVING ALL THE PARAMETER INTO THE SYSTEM. 43
FIGURE 28 PARAMETER STRUCTURE SHOWS IN CONSOLE LOG IN BROWSER 43
FIGURE 29 ANIMATION EVENT IN ANIMATION.JSON FILE 44
FIGURE 30 ANIMATIONADDITIVE EVENT IN ANIMATION.JSON FILE 44
FIGURE 31 CODES OF READING LOCAL .TXT FILES 44
FIGURE 32 CODES OF SAVING PARAMETERS FROM LOCAL .TXT FILES. 45
FIGURE 33 CODES OF BUILDING POSITION DATA IN FORMAT OF VECTOR 3 45
FIGURE 34 CODES FOR CONSTRUCTING ARRAYS. 45
FIGURE 35 CONSOLE LOG OF THE DATA WITH THE SEQUENCE OF TIMESTAMPS 46
FIGURE 36 START ANIMATION INTERFACE 47
FIGURE 37 CODES FOR STOP ANIMATION BUTTON 47
FIGURE 38 ANIMATION SPEED ADJUSTMENT 48
FIGURE 39 VALUE OF THE SLIDER WORKS ON THE SIMULATION TIME 48

Figure 40 Simulation Time shows in the Scene. 48
Figure 41 Simulation Clock Paused 49
Figure 42 Simulation Clock Stopped 49
Figure 43 Animation Events in Animation. JSON File 50
Figure 44 Definition of Animation Function in Babylon Library 50
Figure 45 Codes of creating Animations 50
Figure 46 Codes for realizing "placementRelTo" 51
Figure 47 Codes of setting Animation Keys 51
Figure 48 Codes for relating Keys with Animation 52
Figure 49 Codes for executing the Animation. 52
Figure 50 (a) .txt File for AnimationAdditive Events (b) .txt File for Animation Events 52
Figure 51 Codes for setting Keys for AnimationAdditive Events 53
Figure 52 Show and Hide Functions 53
Figure 53 Hide and Show Events in Animation.JSON File 54
Figure 54 Codes of "placementRelTo" in Show Events 54
Figure 55 Information of changing Model in State Event in Animation.JSON 55
Figure 56 Imported Mesh (red circle) 55
Figure 57 Mesh State Section 56
Figure 58 (a) All Meshes selected (b) Specific Meshes checked 56
Figure 59 Structure of State Description Section 57
Figure 60 Functions for Checkboxes of State Events 57
Figure 61 Codes for selecting Meshes by clicking on the Mesh. 58
Figure 62 Codes for finding and saving checked Checkboxes. 58
Figure 63 Codes for executing the State Events 59
Figure 64 Description of States during the Animation 59
Figure 65 Trail Events in Animation.JSON File (a) Turn on the Trail (b) Turn off the Trail 60
Figure 66 Codes for creating the Trail in Babylon 60
Figure 67 Codes for turning on and off the Trail 60
Figure 68 (a) No Mesh picked (b) Workstation is picked (c) Pallet is picked 60
Figure 69 Corresponding Checkbox is checked when the Mesh is picked. 61
Figure 70 Test of the GUI Functionalities 62
Figure 71 Test of Mesh State Functionality 63
Figure 72 Screenshot of the Animation 63
Figure 73 The last Moment when the Animation is finished. 64
Figure 74 The Screenshot of the Moment when the Pallet is in the Buffer. 64

LIST OF TABLES

TABLE 1 ADVANTAGES AND DISADVANTAGES OF EACH SOFTWARE TOOLS [30] [31] [32] 18
TABLE 2 COMPARISON AMONG 3 SOFTWARE TOOLS[33], [34] 19
TABLE 3 COMPARISON BETWEEN UNITY AND UNREAL ENGINE[44], [45],[46] 21
TABLE 4 COMPARISON OF THREE 3D WEBGL ENGINES[50], [51] 22
TABLE 5 TABLE OF PARAMETERS OF WORKING STATIONS 33
TABLE 6 TABLE OF PRE-DEFINED CONSTANTS 35
TABLE 7 LINKS TO MESHES IN DATABASE 36

CHAPTER 1. INTRODUCTION

With the development of technology, the manufacturing systems become more and more complex which makes the decision-making of production system designs and production plans take more time, more costs, and less efficiency. Thanks to the idea of digital twin and virtual manufacturing, the virtual factory has been pushed forward and been developed to make decision-making more efficient and more economic. However, although with the development of the virtual factory, there are already several software tools in the market, some of them still have their own weaknesses and problems which the users might meet during the operation. In this paper, a new solution has been introduced according to the project called Virtual Learning Factory Toolkits(VLFT) which decouples the software tools into three stages according to three different functionalities during the system simulation which are simulation, data processing, and 3D visualization. By developing the algorithm in the data processing stage and a new application in the 3D visualization stage, the solution achieves more compatibility among the software tools and operating systems.

In this chapter, the first section is an introduction of the background of the virtual factory and an analysis of the weaknesses and problems in the existed virtual factory software tools which leads to the second section of setting the aims of this paper to solve parts of these weaknesses and problems. The last section introduces the whole structure of this paper and the contents of each chapter.

1.1 BACKGROUND

As products, processes, and data are becoming more and more complex in manufacturing due to more and more focus of the market on the delivery of customized and diversified products, quality improvement, and time reduction. The decision-making process takes more and more time to take into consideration all kinds of variables. However, to shorten the delivery of the products, quickly respond to customer's needs, react to the market changes, and making correct decisions as earlier as possible is the only way to make companies succeed in nowadays market.

On the other hand, once a design of a production system has been released and in its production phase, a change in the system might cause a large cost in manuals, processes, training, government approvals, etc which leads to the increases of the cost of the products. Therefore, a precise conceptual design that takes all the facts into consideration is needed to reduce the unnecessary cost for the companies.[1]

To consider all factors mentioned above, the manufacturing system has been divided into three stages following the timeline of the evolution of the manufacturing system according to Zhou et al.(2018) which are traditional manufacturing system, smart manufacturing system, and intelligent manufacturing system.

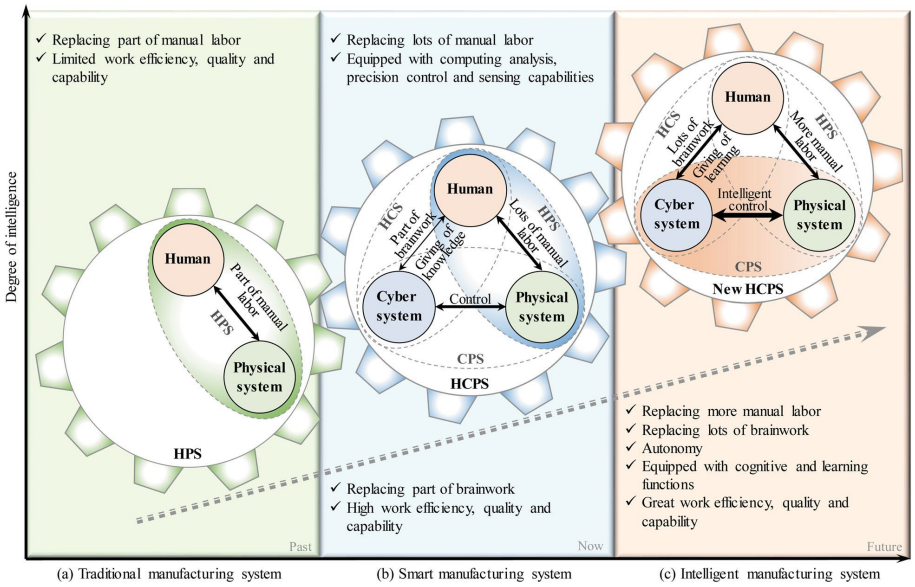


Figure 1 The Evolution of Manufacturing System[2]

The traditional manufacturing system consists of humans and the physical system. the physical system can replace part of manual labor but the restriction of this kind of system is that humans should have high ability in information perceiving, analysis, decision-making, operation, and control. It is quite time-consuming, high labor intensity, and high risk when the decision which has been made goes wrong.[3]

Comparing with the traditional manufacturing system, the smart manufacturing system adds the part of the cyber system and more based on the internet technology. Thanks to the cyber systems, the smart manufacturing system replaces part of brainwork in addition to replacing part of manual labor. It can help people to do part of perceiving, analysis and decision-making. The efficiency, quality, and stability of the smart manufacturing system are markedly improved. The key issue in smart manufacturing is the convergence of physical systems and cyber systems, where digital twin, a reference model to realize the interoperation and fusion between physical space and virtual space of manufacturing, is regarded as the most effective solution nowadays.[4], [5]

Intelligent manufacturing systems are the trend of development for the future which can replace much more brainwork than smart manufacturing systems. It represents an in-depth integration of new-generation artificial intelligence technology and advanced manufacturing to obtain powerful learning and cognitive capacities. The key issue in intelligent manufacturing is the development and applications of knowledge-based intelligent skills.[6], [7]

Nowadays, most of the manufacturing system is in the second stage which is the smart manufacturing system, and more and more companies start to use the simulation technology, software tools, and digital model to reduce the costs and increase the profitability. Moreover, by following the big trend of Industry 4.0, the idea of digital twin has been used more and more frequently in the industrial area for realizing the concept of smart factory which can create a virtual copy of the physical world and make decentralized decisions.[8]

The first definition of DT given by NASA as “an integrated multi-physics, multi-scale, probabilistic simulation of a vehicle or system that uses the best available physical models, sensor updates, fleet history, etc., to mirror the life of its flying twin. It is ultra-realistic and may consider one or more important and interdependent vehicle systems”[9]. Digital twins can integrate the physical and virtual data throughout a product lifecycle, which leads to a huge volume of data that can be processed by advanced analytics which enable manufacturers to make more accurate predictions, rational decisions and informed plans. According to the categorical review on DT in the manufacturing area, DT can be classified into three parts in terms of different integrations which are digital model, digital shadow and digital twin. The difference between each other is based on the level of data integration which shown as the figure below.

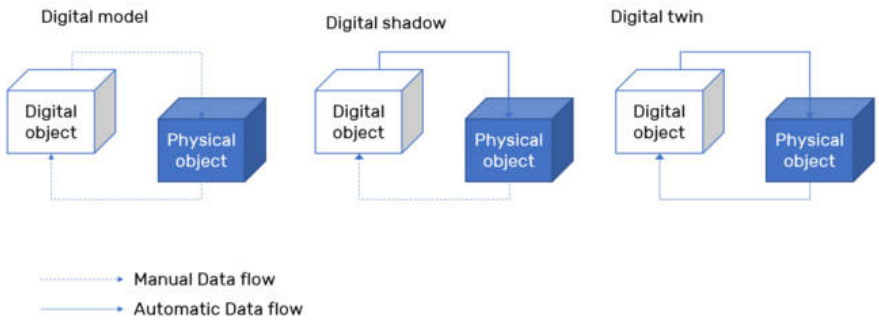


Figure 2 Data Flow in (a) Digital Model, (b) Digital Shadow, (c) Digital Twin

In the industry area, the idea of digital twin leads to the term calls Virtual Manufacturing (VM). Different from the traditional method of designing a manufacturing system, designing a manufacturing system with Virtual Factory based on the idea of VM allows, 1) to compare different management decisions from different specific states during the performance evaluation of the factory, 2) to reconfigure the existing system in a short time to keep the flexibility of the system to fit different production plans, 3) to support ramp-up phase of production system when it is still incomplete in the design phase. [10] 4) to improve workers efficiency and safety by using the virtual production system during the training and learning, 5) to reduce the production times and material waste when setting the plan for the new products, 6) to create a collaboration network among people in different cells or job shops working on the same product. [11]

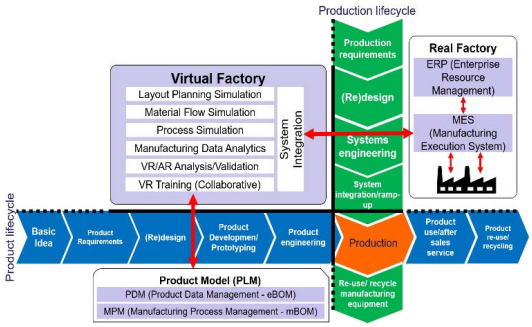


Figure 3 Digital Twin based Virtual Factory Concept[12]

In nowadays market, there are several commercial software tools (e.g. Arena from Rockwell, Plant Simulation from Siemens, Simio, etc.) providing an integrated environment to describe the system and its behavior in terms of relationships and rules, dealing with the generation of random values and the collection of the statistics for building a simulation model of a manufacturing system.[13] However, although all of these software tools are quite strong, there are still some problems and weakness need to solve which are,

1. Almost all the simulation software tools need to be installed before launching and normally the installation of the software tools needs the license which might need some cost for the acquisition. It could be a problem if for example there is someone have received a project which must be opened in a software tool, he/she doesn't have, so he/she needs to spend some time and maybe also some money to get the software tool first, it will cause the inconvenience for the people who work on the same thing but with different software tools.
2. Also, not all of the simulation software tools support all kinds of operating systems. For example, Arena from Rockwell doesn't support Mac OS and Linux. And Siemens PLM software as well.
3. Most of the simulation software tools are too complex for the beginner who needs a tutorial before using it and also in the software tools, there are a lot of professional terms which require the educational background of the user. So, if there is someone who is a beginner in the industrial field, using these software tools could cause problems and confusion for him/her.
4. As mentioned in problem 1, there is always a weak link between two software tools which can be a problem for different PC to open.
5. Another problem is all of these software tools have a limited model library which makes the virtual model not as same as the system in reality in some cases. In another word, users can not import their customized model into the system if their desired model is not existing in the library.
6. In most of the software tools, to launch the animation of the manufacturing system always need a long time to do the preparation which is not suitable for the case if someone want to demonstrate the animation of the system during meeting or presentation.

To solve all the problems have been stated above, in this paper, a new browser-based application is introduced. The new application runs in the browser which means there is no requirement of installation. The main reasons to build a browser-based application are,

1. To let all users can open the simulation model with different operating systems.
2. The application itself is free and easy understanding for both the professional workers and the beginners.
3. Allows users to build and import their customized models. (integrate the data sources, coherent with industry 4.0 and digital twin, help to do better exploit, multiple users(briefly))
4. Instant animation which allows users to do the demonstration easily and quickly.

1.2 THESIS AIMS

This thesis focuses on 3 aspects of the new developed browser-based application which are the development of the graphical user interface (GUI), the functionality of the animation for the simulation model and to build a link between the simulation software tool with the new application.

The result of this thesis covers most of the problems which have been stated in the previous section and to achieve those aims by carrying out the following process:

- Identify the information needed to provide the described dynamic representation.
- Continue to develop the application VE.bjs based on the version has been done by the previous colleagues and program the functionalities of the simulation animation.
- Adjust and add necessary additional functionalities for users to get more information during the animation executing.
- Use a discrete event simulation environment (Java Modelling Tool) to model the required information and generate logs of the dynamic evolution of the factory.
- Use Python to generate log to provide a coherent representation of the evolution of the factory.

1.3 STRUCTURE OF THE THESIS REPORT

This thesis report is organized in seven chapters, references and a final section with appendices includes the full version of codes in both JavaScript and Python. The Chapter 1 is the introduction of the background of the topic of the thesis, the thesis aims and the structure of the report. In Chapter 2 is the state of art of this thesis paper which includes the literature review in virtual factory area and a technical research in the software tools related to the virtual factory topic. In Chapter 3 is the statement of problems have been done according to the conclusion from chapter 2, the requirements, goals, boundaries and possible users. Chapter 4 describes the method and approach to solve the problems has been stated in the chapter 3. Chapter 5 introduces the solution to realize the approach in detail. Chapter 6 contains the experiment and validation have been done for validating the result. The last chapter, Chapter 7, states the conclusion and future potential improvement of the application.

CHAPTER 2. STATE OF ART

This chapter covers the literature reviews and technical research of the topics which are related to this paper. At the beginning, the section about virtual factory is mainly focusing on the development of modelling and visualization technologies of the virtual factory. Then, available on the market discrete event simulation software tools have been discussed and the comparison among three most commonly used software tools have been done. In the third section, both the virtual reality software tools and 3D WebGL engines in nowadays market has been described with the comparisons among the most widely used software and engines.

2.1 VIRTUAL FACTORY

In 1993, the virtual manufacturing concept, which is integrating product and factory models as a critical aspect of virtual factory has been introduced by Onosato and Iwata.[14] According to Jain et al. the definition of virtual factory could be: "as an integrated simulation model of major subsystems in a factory that considers the factory as a whole and provides an advanced decision support capability."[12] Nowadays, with the rapid development of the industrial technology due to the needs to deal with more and more complex production system, a new idea of insert new digital and virtual technologies in the manufacturing sector becomes more and more popular. In this context, the virtual factory paradigm can assist to respond to this need for innovation by addressing various key issues:

- Reduction of production times and material waste thanks to the analysis of virtual mockups of new products.
- Development of a knowledge repository where people can find any kind of stored material(design or documents) in different versions.
- Improvement of workers efficiency and safety through training and learning on virtual production systems.
- Creation of a collaboration network among people concurrently working on the same project in different places.[11]

2.1.1 Virtual Factory Modelling

The benefits of virtual factory and virtual manufacturing consist of the multi-layer integration of (1) various activities along the factory and product lifecycle; (2)hardware, software tools and human resources; (3)real world and virtual world.[14] By integrating these information, modelling and simulation approaches can be widely used to support and speed-up decision making not only in the design stage but also in managing the process. However, the information from the real factory should be synchronized with the virtual factory which leads to several problems which have been stated by B.Kadar et al.[15] are:

- Data acquisition and validation for simulation input.

- Quick response time of simulation runs and analysis.
- The ability of creating the snapshot of physical system status in the simulation model by instantaneous feedback.

The possible solution of these problems could be data standardization. For the standardization efforts, ANSI/ISA-95 is an international standard for developing an automated interface between enterprise and control systems. However, Although ANSI/ISA-95 enables user to freely define customized properties that can attached to most of the classes representing processes and production resources, it doesn't provide a complete support for modelling physical data in the manufacturing system.[13]

Based on the Standard for the Exchange of Product model data (STEP), the Industry Foundation Classes(IFC) is a platform neutral, open file format specification which the data model intended to describe architectural, building and construction industry data. It is a commonly used collaboration format in Building Information modelling(BIM) based project.[16] The IFC standard was mainly conceived for Architectural Engineering Construction(AEC) industry domains which provides most of the definitions needed to represent tangible elements of a manufacturing system.[13]

Another approach which is ontology-based approach given by National Institute of Standards and Technology(NIST) is Process Specification Language (PSL)[17]. The PSL is a set of logic terms of describing processes and they are specified in an ontology that provides a formal description of the components and their relationships that make up a process. The ontology merges the information of concepts such as objects, activities, timepoints, etc., constants, functions and relations of the process and using the Common Logic Interchange Format (CLIF) to represent these concepts.[13], [18]

2.1.2 **Virtual Factory Visualization**

Visualization has become an important tool in many areas for explaining and understanding large and complex data. It can combine machine intelligence with human intelligence to support decision-making under complicated scenarios. According to the survey has been done by Zhou et al., a classification of the application scenarios has been done which based on the concepts of 'replacement' and 'creation'. The replacement means to liberate people from dull work by replacing through smart devices and create a virtual environment of complex and dangerous work scenarios in computer to let people learn skills in a safe and informative way. Meanwhile, the creation means to develop a robust visualization application to present both the summary and details of various manufacturing data, and provide users with interactive manners to facilitate interesting pattern identification and valuable information exploration, hence supporting human-in-the-loop decision-making in a wide range of innovation and creation scenarios. [19]

By using the immersive technologies which can be classified into virtual reality(VR), augmented reality(AR) and mixed reality(MR), they blur the boundaries between the physical world and the digital world which makes possibility for people to familiarize the complex production process even when they are not in the factory.[20]

By following this idea, many solutions and research has been done in the past decades years to develop the virtual factory technologies in order to make the decision-making of the production life more

efficient. Wolfgang et al. have used CASUS(computer animation of simulation traces) system to build and visualised the behaviour of an electronics assembly industry based on discrete events simulation[21]. Kadar et al. have developed a 3D software tool(GIOVE) to design the production system and place the system inside the virtual factory[22]. Zhou et al. conducted an animated simulation with the 3D engine Unity3D to explain how hot gases escape form the door of reheating furnace in the production of flat-rolled steel products and deepen experts' understanding of the internal state of reheating furnace. Also, many leading manufacturing companies has implemented the visualization of virtual factory, for example, Ford Motor Company improves assembly line performance in its European facilities by evaluating and optimizing the designs using virtual factory system[23]. Siemens have developed the applications for the virtual factory Siemens PLM software tool[24].

2.2 SOFTWARE TOOLS OF DISCRETE EVENT SIMULATION

With the rapid development of the technology related to the idea of Industry 4.0, virtual factory and digital twin, in the market there are already several mature simulation software tools such as Arena by Rockwell Corporation, Flexsim by Flexsim Software, SimEvents by MathWorks, Anylogic by Anylogic Company, Plant Simulation by Siemens PLM Software and so on. All of these software tools have their own highlights and drawbacks. In this paper, three most popular software tools which are Arena, Flexsim and Anylogic have been chosen for doing comparison to find out their common pros and cons. But first of all, in the first section, the challenges of the software tools of virtual factory facing in nowadays market will be stated as a guideline for choosing software tools.

2.2.1 Virtual Factory Software Tools Challenges

By following the idea of the digital twin, before developing the software tools to realize the virtual factory, a list should be defined which lists major challenges that a manufacturing company has to face to effectively implement it. According to the challenge defined by Gianfranco E.Modoni et al.[25], they could be :

1) Connection with the real factory. It means the data acquisition, validation and transmission in the software tools should be reasonable enough that whenever the acquisition of real measurement is not feasible, the data from the software tools can be used reliably.

2) Granularity of the synchronization process. The granularity of the synchronization process affects the distinguishability of the model from its real counterparts and the processing costs. Now the enabling technologies related with this challenge include multi granularity/scale data planning and High-Performance Computer. [26]

3) Management of the real-time and historical data. It manages mainly the capability to update the decision of maintenance strategy, to predict failures and to implement product lifecycle energy optimization strategies. The challenges are related to the data transferring between DT and real factory in both directions which are the mechanisms for managing data leveraging databases and microservices to historicize and display the acquired data and cutting-edge processing and storage mechanisms operating on cluster system distributed on cloud.

4) Supports for advanced simulation and forecasting tools. DT should both enables to monitor information in real time and collect and use this information to properly simulate the factory performance. To realize this, DT should fully support by the various software tools that enable to model and simulate the complex dynamics of a manufacturing system and should set up the initial conditions of simulator of the simulation models through a true snapshot of the real system[27]. The software tools should contain these technologies which are multi-scale modelling technology, virtual production operation simulation and verification technology, virtual maintenance technology, virtual reality and augmented reality technology to meet with the challenge.[28]

5) Data and intelligence distribution. The challenge is for the study of valid strategies for distributing intelligence and data of the DT close to various production resources to avoid redundant data from the shop floor to the DT. The possible solution could be Edge Computing paradigm[29] and the microservice based architecture.

6) Enhancing the interoperability of the production resources. The concept of DT requires a homogeneous perspective of the handled information that persists across the different functional boundaries. But among the software systems, there is always lack of compatibility. By solving this challenge, the DT system could let more participants work on the same project without any misunderstanding and all the resource can be shared and updated.

2.2.2 **Comparison of discrete event modelling simulator in the market**

As mentioned before, in this paper, a comparison has been done among three most popular discrete event modelling simulator in the market which are Arena , Flexsim and Anylogic with the generated information from several research documents.

First of all, before doing the comparison, the key features, the advantages and disadvantages of each software tools will be state in the following table.

Table 1 Advantages and disadvantages of each software tools [30] [31] [32]

	Arena	Flexsim	Anylogic
Key Features	1, Flowchart modelling methodology. 2,Realistic 2D and 3D animation design. 3,Reduce operating costs. 4,Increase profitability through operations. 5,Reduce delivery time. 6,Diagnose and eliminate bottlenecks	1, Manufacturing: job shop, assembly line. 2, Logistics and distribution: supply chain design, storage layout. 3, Transportation: system traffic flow, custom traffic congestion. 4, Material handling: packaging and warehousing. 5, Networking data flow. 6, Mining processes.	1,Multimethod modelling environment (Discrete Event, Agent Based and System Dynamics). 2,3D and 2D Animation and visualization 3, Industry-specific libraries. 4, the only professional software tool for building Professional agent-based simulation software. 5, GIS maps integration. 6, A full range of cloud technologies. 7, Rich experiments framework.

Advantages	1,Good in quickly setting up simple simulation models without animations. 2,More emphasis on logic and process flows. 3,Easy for the beginners and people don't have industrial education background. 4,Complete range of statistical distribution.	1, Flexsim jumps straight to animating the process and the quality of the animation is good. 2,Have more visual expression during building models. 3, Don't need to separate logic flow with animation. 4, Easy for beginners and people don't have industrial education background.	1,Anylogic is JavaScript based. 2, mainly focusing on coding and programming the process. 3, A good choice for developers who can tweak background parameters. 4, Better animation and graphics(compare with Arena). 5, Being able to simulate a huge and complex model.
Disadvantages	1,Plain graphics. 2,Bad in advanced animation. 3,Bad in simulating complex models.	1, Need to go through more clicks before understanding how statistics is drawn out of the model. 2, Not a good choice for simulating complex models.	1, Not suitable for non-developers or coders. 2, Need long time to learn.

According to David Unrau[33] and the website descreye.com[34], these three software tools can be compared in 5 main aspects which are 1)capability, 2)integration and extension, 3)support and documentation, 4)usability and 5)price and each of the aspect will be divided into detailed factors. The measurement of the comparison will be Best, Better, Ok, Worse and Worst.

Table 2 Comparison among 3 Software Tools[33], [34]

		Arena	Flexsim	Anylogic
Capability	Execution Speed	Better	Best	Better
	Optimization Capability	Better	Better	Better
	Distribution Fitting	Better	Better	Better
	Programming	OK	Best	Better
	Database Connection	Better	Better	Ok
	Graphics	Worse	Best	Ok
	Default Library Quality	Ok	Better	Ok
	Customization	Better	Better	Better
	Statistics Capturing	Better	Best	Ok
	Pre-saved Objects	Better	Better	Worse
	Documentation	Worst	Worse	Worse
Integration and extension	Compatibility to Database	Ok(through ActiveX data object files)	Worst	Ok
	Programming Language	Unknown	C++ & Flexscript	Java
	Learning Resources	Ok	Ok	Ok

Support and documentation	Support Operating System	Windows	Windows	Windows, Mac OS X, Linux
Usability	Training Requirements	Ok	Ok	Ok
	Frequency of Programming	Not Frequent	Not Frequent	Frequent
	1st Model Building Time	Longer	Shorter	Longer
	3D or 2D	Both	Both	Both
Price	One Year Licence	$2495USD (Arena Basic)	$20000USD	$15800USD(Professional)

According to the table above, all these three software tools are quite good in executing speed, optimization capability, distribution fitting and customization. Also, all of them have both 2D and 3D models. However, Arena is weak at graphics representation, documentation supporting and long time to get started. Meanwhile, documentation supporting for Anylogic and Flexsim is also not well and Flexsim has problem with connection to database. For the operating system, only the Anylogic supports other operating systems apart from Windows and the rest two only support Windows system which might cause inconvenience for the Mac users and others. Another factor is the price, all of these 3 software tools need to spend a lot of money annually which might cause problem for model sharing, start-up and some other events.

2.3 VIRTUAL REALITY IN MANUFACTURING

The virtual reality is a well-adapted mean to help people to interact with Cyber-Physical Production System. It offers 3D models, realistic rendering and collaboration functionalities.[35] Now in the market, there are three kinds of virtual reality which are desktop system, wide-screen projection system and immersive VR system using HMDs[36]. The desktop system is the most popular one for now which has been used in a wide area in the daily life[37]. In the manufacturing area, for the virtual factory software tools which have mentioned before, all of them are kind of non-immersive VR which provide well-rendered 3D model and virtual factory environments.

The VR technology in manufacturing system can be used in workplace design and layout, equipment design, training, product development, review of workflows and benchmarking, quality assurance and risk management and control of robotics[38], [39]. By applying VR technology in manufacturing system, the benefits could be improvement of employees' safety, manufacturing better products and making plans in a safe place[40], [41]. However, there are some hinderances to use of virtual reality in manufacturing which are the cost for the developing the software tools and the integration which might not easily to integrate with manufacturers' existing design systems. [39]

2.3.1 Virtual Reality Gaming Engine

In order to solve those problems and to make easy for people to develop and operate with the VR software tools. Nowadays, more and more industrial projects try to build the virtual environment and the virtual factory with the VR gaming engines due to their perfect 3D graphical rendering, partially open source code, networking capabilities which can serve for multiuser simulation and the price is relatively inexpensive.[42], [43]

In the market, there are several 3D gaming engines which are Unity, Unreal Engine, Godot, CryEngine, Amazon Lumberyard and so on. Among these engines, Unity and Unreal Engine are the most used gaming engine not only for gaming but also for other industrial usages. In the following table, a comparison between these two most popular gaming engines has been done considering all the aspects based on the views of industrial requirements and the quality of graphical user interfaces.

Table 3 Comparison between Unity and Unreal Engine[44], [45],[46]

		Unity	**Unreal Engine**
Visual		Impressive of its High-Definition Render Pipeline.	Generally better due to packed with tools and presets.
Versatility	2D& 3D Model	Good at toolset.	Better at graphical fidelity.
	VR	The plugins are very versatile and integrate into the overall XR infrastructure.	
	Multiplayer	Requires 3rd party tools support.	Integrated support.
Learning Curve and Ease of Use	Programming Language	C#	C++
	Documentation	Richer	Developing
	Community	Has larger community which has more examples	
	For Starting-up	Easier for the beginners	A bit overwhelming that multitude of available options and poor segregation make navigation difficult.
Price		Free	Free

According to the table above, both of these two software tools are good at graphical visualization and no requirements for payment. However, there are still some weaknesses for each software tool need to be improved.

2.3.2 **WebGL**

Another option is to develop a VR model in browser without requirement for installation with the JavaScript API called WebGL. Web Graphical Library(WebGL) is a JavaScript API for rendering interactive 2D and 3D graphics within any compatible web browser without the user of plug-ins. The WebGL is fully integrated with other web standards, allowing GPU-accelerated usage of physics and image processing and effects as part of the web page canvas.[47] The reason for the appearance of WebGL is that until a several years ago, the interactive 3D graphics had a poor availability on the Web which makes even the most robust and most popular browsers can't be utilized for a full 3D graphics presentation and interaction and none of the web browser had built-in 3D graphics logic for presentation that will not require a plug-in to the browser.[48]

However, after the WebGL has been introduced in 2011, the browser became capable of using hardware accelerated graphics without plug-in such as Flash or other out-source applications. The WebGL is one

of the most developed graphic APIs which is supported by HTML5 and by extending JavaScript, the 3D content creation and rendering on the web are been supported.[49]

2.3.3 3D WebGL Engine

There are several most-used 3D WebGL engines in the market which are Babylon.JS, Three.JS, PlayCanvas and so on. Most of these 3D WebGL engines are normally used for gaming thanks to their perfect rendering and fluent animation representing and these are the reason why the new-built application has been developed with these kind of 3D WebGL engines. A comparison between Babylon.JS, Three.JS and PlayCanvas has been done in the following table which consider the aspects of the power of the engine, compatibility and the support of the engine.[50]

Table 4 Comparison of three 3D WebGL Engines[50], [51]

	Babylon.JS	**Three.JS**	**PlayCanvas**
Online Editor	Yes	Yes	No
Library	Large	Large	Large
GUI Tools	Yes	No	No
API Documentation	Lack	Good	Yes
Examples	Plenty	Plenty	Lack
Support	Strong	Lack	Lack
Community	Large	Large	Sometime Private
Performance	Good	Great	Good
Game engine	Yes	No	Yes
Rendering Pipeline	Yes	No	Yes

All of these three engines have their own pros and cons. For the Three.js, it is easy to learn, has plenty of examples, has large community, good documentation and great performance, but there is no rendering pipeline inside the engine which makes a lot of modern rendering techniques impossible to implement, also it lacks support which might cause bugs for the users.

For the Play Canvas, it is an open-source tool which the platform is mobile-first. It has zero compile time which can make process faster and inside the engine, it has integrated physical engine. But the cons of Play Canvas are that some of the projects are private which need to pay, also the examples are not too much that the tutorials for Play Canvas are few and far between.

For the Babylon.js, it has its own playground which is a great tool for testing. It has strong community support and maintenance which is helpful for the user and react to bugs very fast. It also has Up-to-date codebase which is convenient for user to use and the PBR rendering is excellent. However, the weaknesses of Babylon.js are that in Babylon.js, the documentation is not plentiful, and since the first version of Babylon.js released in 2013, so the engine is still young which is lack of maturity.[51]

CHAPTER 3. PROBLEM STATEMENT

This chapter conclude the problems which has been found according to the statements in the previous chapter and according to those problems to set the requirements, goals and boundaries when giving out the possible solution. The last section of this chapter discussed about the possible users which could also be set as a goal for the developers to achieve.

3.1 PROBLEM STATEMENT

According to the analysis of the discrete event simulation software and the virtual reality software in the previous chapter, there are some common problems among these software tools need to solve.

From the view of discrete event simulation, the potential problems could be:

1. Poor graphics representation like Arena which can't represent the real factory in the virtual environment.
2. Poor document support like Arena which might not be user-friendly.
3. Limited kinds of support operating system like Arena and Flexsim which might be inconvenient for users.
4. Poor compatibility among different software which limits the choices for users.
5. High price which might not be friendly to personal users.
6. Requirements of installation time which might be a problem for the user to use in some emergency situation.

In another hand, from the view of Virtual Reality software, the possible problems could be:

1. Like for unity, the multiplayer need to be extra-installed.
2. For setting the animation in for example Unity, it requires a long time to learn how to set the relationships between each model and has several scripts need to write which might be a problem for the user who is beginner of programming languages.
3. Although both of them are free but still require installation which means they could not be instantly used.

3.2 REQUIREMENTS, GOALS AND BOUNDARIES

After stating all the possible problems according to the conclusion from the state of art, before starting to solve those problems, requirements of the hardware and software, the specific goals for solving those problems and the possible boundaries should be stated. In this section, the possible requirements, goals and boundaries has been listed.

3.2.1 **Requirements**

The possible requirements for solving these problems could be:

- Following the concept of virtual factory, the simulator selected should be robust enough to include most of the industrial behaviours and scenarios which happens in the real factory.
- The software tool should have good 3D graphical representation which can create and render a realistic virtual environment. Also, a better GPU in PC is possible required to have a better rendering quality.
- If the software tool is browser-based which developed with WebGL, a browser which support the WebGL API is required.
- The software tool should have a strong document support and possible to have a huge community which can be helpful for users to learn and solving problems during operating with the tools.
- The software tool should also support different operating systems to meet the requirements of all kinds of users.
- Compatibility of the software tool should also be considered. The tool should be compatible with other software tools to let users have more choice of tools during the operation.
- If it is possible, the price of the software tool should be free or low which most people can accept and also it is better to have no requirements for installation so that people can use the tool as soon as possible.

3.2.2 **Goals**

According to the problem statement in the previous section, the goals can be set as:、

- To develop an application which can represent most of the industrial behaviours and scenarios in a good 3D graphical representation.
- To develop a strong documentation and community of the application which users can find all kinds of supporting materials.
- Possible development of the application with WebGL API which only have a weak restriction of the browsers and can be operated in all kinds of operation system.
- Develop the application, which is user and beginners friendly, for example, to build a clear graphic user interface.
- The application should be open source which can let users to customize the model inside freely.
- The application should be compatible with most of data types from outsource.
- The application should be easier to install or no requirements for installation.

3.2.3 **Boundaries**

The boundaries of the possible development of the application could be divided into several aspects which are hardware restriction, software restriction and also it might be the network protocol.

For the hardware boundaries they could be:

- The RAM capacity which may affect the quality of the animation.
- The GPU capacity which may affect the rendering of the 3D virtual scene and the fluency of the animation.

For the software boundaries they could be:

- Due to the programs structure, for running the animation, the nodes of animation might be limited due to the time for reading and parsing the data from imported file.
- For representing the 3D objects, the types or the kinds of the objects will be limited according to the types or kinds of objects in the local folder or in the online database.

For the network protocol, the boundaries could be that some of the applications require permission to read the local files or the permissions to access the database or some specific path.

3.3 POSSIBLE USERS

Among all the requirements what have mentioned before, one of the most important points is that the application which will be developed should be user-friendly and fit the expectation of users for this application. So, first of all, what kinds of people will use this application should be discussed in case to figure out what are those people's demands for the application.

There are two categories of possible users which can be classified as human users and machine users. For the first category, it can be divided into four main types of users. They are professionals, top managers of companies, workers in the factories and students who study related subjects. The new application which based on the idea of virtual manufacturing works with different purposes according to these different types of users.

1) For professionals, the main benefits by using this new-developed application are related to the research and designing of a production system. It allows the professionals to do the research or designing of production system without build a prototype but only build a model in a virtual environment which cuts down the major cost for production and research but gets the same or even more accuracy for the analysis which covers all kinds of parameters in the system.

2) For the top manages, the application can be used to quality management and risk management. It can be used to conduct a more comprehensive quality assurance checks and manage risk by keeping all parties accountable. It might be used for periodic review of products.

3) For the workers in the plants, the application can be used in two ways. The first case is when a new production system has been built or the optimization of the production system has been

done, the application can help workers to get used to the new system in advance which can save the warm-up time. Also, it is helpful for training the new workers before they work in the scene. The second usage is to let the worker review workflows and improve benchmarking processes. Workers can see what's going on , review how workers' responsibilities could be changed and determine if the changes would save the manufacturer money.

4) The last types of human users could be the students who study in related fields. The application can help students to understand the production systems in factories through a more intuitive way which is useful for the beginner of the industrial field to understand the machines, the flow lines and other facilities and behaviours in the factory. Also, since the application is easy to operate, students can work on the application by themselves and to learn from designing a production system by themselves or setting specific parameters in a existed model.

The second type of user is machines. The application can be used mainly as a tool for the machine learning. The simulation model can be built in the application and get data by running the simulation. Then based on the analysis of the system, the machine can automatically learn to do the diagnostic, predictive and prescriptive analysis.

Also, another possible usage could be that to eliminate the need for advanced control panels when robots are used. This could help enhance the precision guidance of robotics, enabling more cautious and detailed actions without actually putting stress on the employee.[39]

CHAPTER 4. METHOD

To solve all the problems, which is common for the virtual factory software tools in nowadays market, has been stated in the Chapter 1 and Chapter 3, and following the goals has been stated in Chapter 3, a new approach has been pushed forward by the project called Virtual Learning Factory Toolkits which is co-funded by the Erasmus+ Programme of the European Union.[52]

In the following sections of this chapter, first of all, the approach has been described in general. It leads to the second and third sections about the selection of the specific tools for realizing the approach. The last section of this part recalls the general approach with details of tools and usages.

4.1 GENERAL APPROACH

With the idea from the project called Virtual Learning Factory Toolkits, all the problems stated in the chapter 3.1 can be solved in the way that to decouple one software tool into three stages with three different applications which are 1)using simulator for doing discrete event simulation(DES), 2)choosing programming languages to do data processing and 3)using a 3D WebGL representation application to realize the 3D visualization.

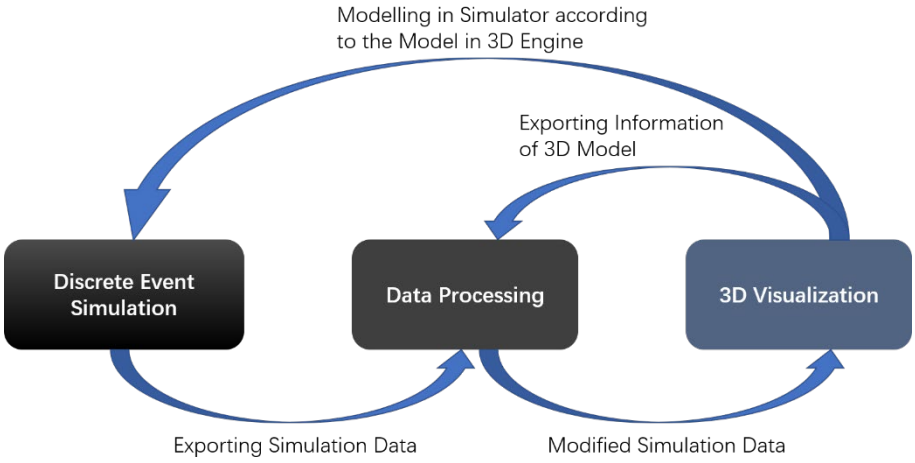


Figure 4 Scope of the Approach

The general scope of the approach shows as the figure above, as mentioned before, three main stages are discrete event simulation, data processing and 3D visualization.

For the DES is mainly about doing the simulation of the manufacturing system and getting data of the industrial behaviors such as processing time, job id, working states, etc.

For the data processing is mainly dealing with the data from both DES stage and 3D visualization stage. The data of industrial behaviors from the DES stage and the information of the 3D entities from the 3D visualization stage will be integrated and analyzed in the data processing stage and translating the data format exported from the DES stage into the format that the 3D visualization stage accepts.

The 3D visualization stage is mainly to represent the 3D model of the system and running the animation of the entities with the data from the data processing stage. Also, user can build models in 3D environment which the information of the entities can be exported and being modified in the data processing stage meanwhile the 3D model can work as a reference for users to build the model in discrete event simulation stage.

By decoupling one software tool into 3 stages, the problem of poor compatibility among software tools can be solved and also it allows users to use different support operating system and different simulator, different programming languages according to their own skill and what software tools they have installed in their computer. In this case, the problem of the prices of the software tools can be solved since people can choose free or already paid applications instead.

To solve the problem of 3D visualization, by developing the application with WebGL, there is no requirement for installation since the application will be browser-based and the API WebGL supports most of the browsers in the market. Then, by choosing a good WebGL engine which has strong capability of 3D rendering, large community and documentations support, user can easily use the application with all kinds of resources.

To build the links between the discrete event simulator with the 3D visualization and to make sure the format of files export from the simulator and import into the 3D visualization application will not have any conflict, the data processing stage is necessary. An algorithm can be developed which accepts all kinds of data format and be able to analysis and transfer the data from one format to another.

For the usages, there are two optional ways for operating among these three stages:

1. First is to build a model in simulator. Then, export the simulation data from the simulator and import it into the data processing stage which will modify and reconstruct into an animation data file represents the situation changes of the entities in the model with the data format according to the format accepted by the 3D visualization application. After building the corresponded 3D model in the 3D visualization application which is same as the model built in the simulator, by importing the exported file from the data processing stage, the 3D animation of the model can run which represents the industrial behavior with the virtual reality.

   This approach can be useful for the people who has designed a manufacturing system and has built a 2D model in the simulator and trying to visualize the system in a virtual environment with 3D models to observe the working state of all the entities at each moment with the animation or present the system to someone else more vividly.

2. Another way to operate is to build the model directly in the 3D visualization application and export the file with the information of the meshes such as position, rotation, scale and the information of the file path. The information from this file can be a reference for defining the mesh file path in the data processing stage and also the 3D model itself can be the reference to build the model in the simulator.

This option provides an opportunity for multiuser who have different duties such as someone is mainly response of doing simulation, someone is mainly doing data processing, and someone is main for building the layout of the manufacturing system. And also, can be useful for an existed system which can be easily built exactly the same in realty in a 3D environment and by exporting the data from the 3D model, to build a model in the simulator to analysis the system and do the optimization of the system.

There could be more operating ways to operate following this scope according to different needs of users by users.

4.2 DISCRETE EVENT SIMULATOR SELECTION

In this thesis, for selection of the model simulator is the Java Modelling Tools. Java Modelling Tools is a suite of applications developed by Politecnico di Milano and Imperial College London. It aims at offering a comprehensive framework for performance evaluation, system modelling with analytical and simulation techniques, capacity planning and workload characterization studies.[53] In JMT, there are 6 different Java applications which can be classified as textual model definition of simulation or analytical, graphical model definition, solution for Markov Chain model, asymptotic solution for model and workload analysis.[54] Its JSIM module is a discrete-event simulator for the analysis of queueing network models. It is possible to define the model inside the framework.[52]

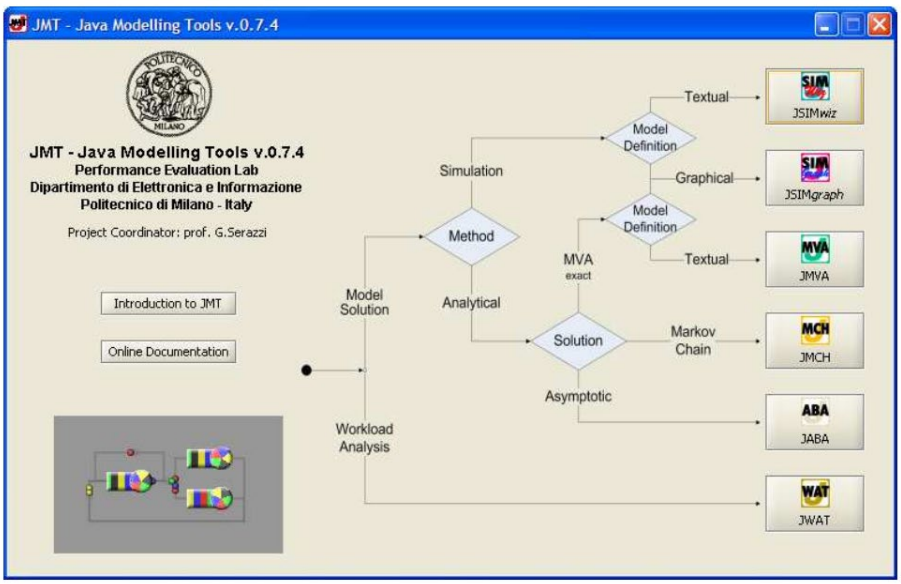


Figure 5 Java Modelling Tools

In this paper, the application has been used is the JSIMgraph(JModel), which offers a graphical user-friendly interface for the simulator engine which indices performance like response times and residence times which are the key performance indices required for generating the animation in this paper and the application enables full portability across different hardware and software platforms(e.g., Linux, Windows, Mac). [54], [55]

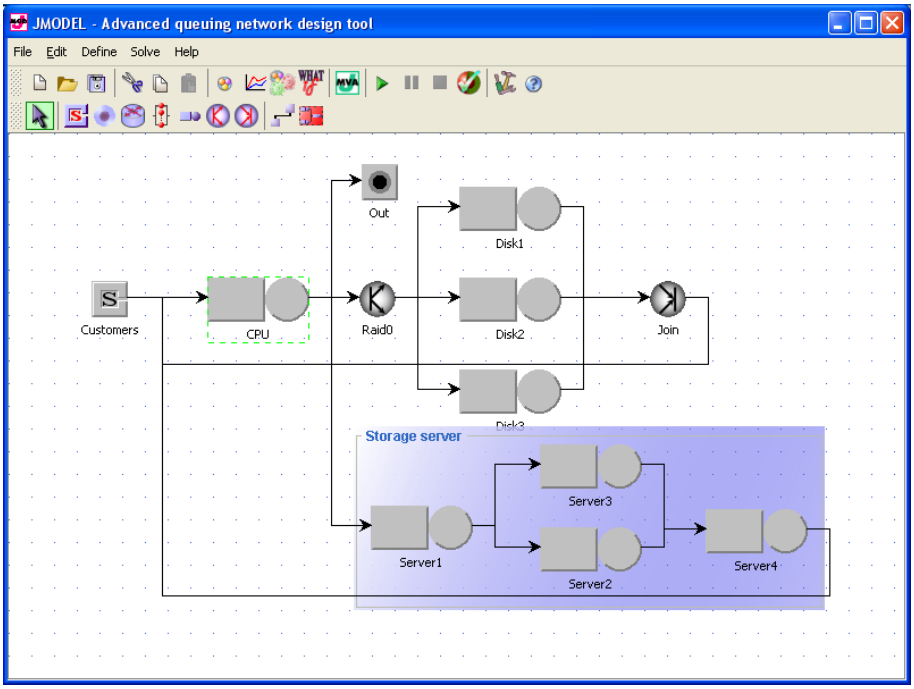


Figure 6 JMODEL[53]

4.3 DATA PROCESSING TOOL SELECTION

As a data processing tool, Python 3 has been chosen. The reason for it is that this programming language is suitable for the fast prototyping, which is necessary for a data processing and transformation. It is a cross-platform tool with a high level of abstraction that makes it perfect choice for non-programmers. Thanks to a great variety of build-in libraries, Python allows processing XML, JSON, CSV and another data formats without having a lot of experience in different data formats.

4.4 3D VISUALIZATION SOFTWARE TOOL SELECTION

According to the comparison has been done in the chapter2, the Babylon.JS is the best option among these three engines although the other two also have their highlights, but in general, Babylon.JS has more advantages in the aspects of powerful supports such as library, examples, GUI tools and large community which allows all kinds of users to share their problems or projects inside the community. Also, another big pro is the online editor in Babylon.JS. Its own playground let the user test codes, share the examples and learn by following the tutorials which makes a significant convenience to the users. By following the goals of the new-developed application of this thesis, the reasons to choose Babylon.JS are also because it is a free engine which is good for both the developer and future users, and it has its own pre-developed scene which is easy for all kinds of users to use, to change parameters of the meshes in the scenes and modify data.

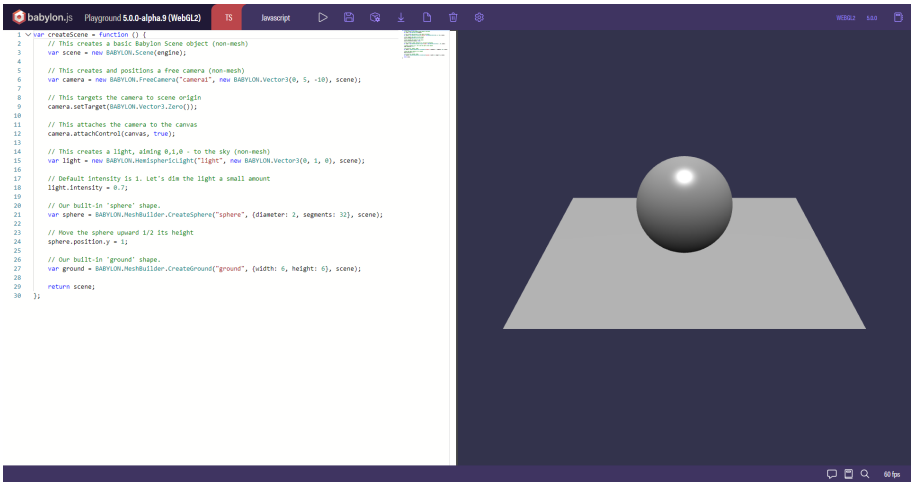


Figure 7 The Babylon.JS Playground[56]

In summary, thanks to all these advantages of the Babylon.JS, in this thesis, the new browser-based application is developed with this 3D WebGL engine.

4.5 Introduction of New Approach

Based on the previous three sections about the selection of the discrete event simulator, data processing software tool and the 3D WebGL engine to operate the application, a full scope of the approach has been built.

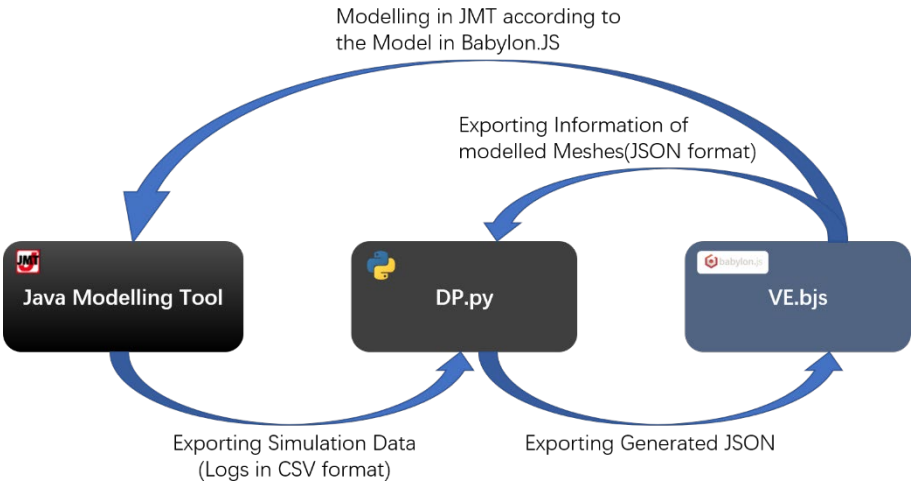


Figure 8 The full Scope of the new Approach

As the figure shows above, the full scope of the approach contains three applications which are Java Modelling Tool, mainly used for building the discrete models and getting the simulation data, DP.py(Data Processing.py), which is a developed algorithm based on the programming language Python, serves for the data processing stage to analyze and translating the data formats. And VE.bjs(Virtual Environment based on Babylon.js) is a reconfigurable model-driven virtual environment application based on Babylon.js which responses for 3D visualization stage.

Thanks to the idea of decoupling one software tool into three, it has more compatibility which for example, in the middle stage for data analysis, DP.py is not the only option for processing but also some other algorithms or applications based on other programming languages such as MATLAB, C++ can do the same functions and so on which depends on different software tools has been installed in the users' computer , level of expertise in a specific programming language and also according to which license of which software tool the users have.

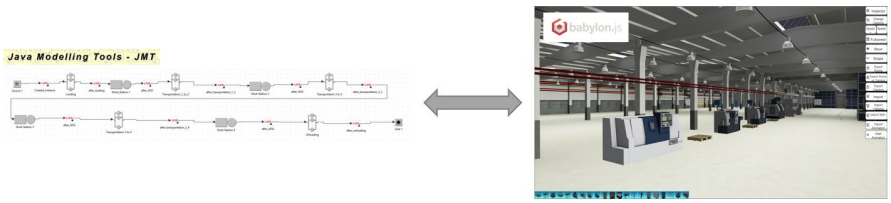


Figure 9 The Scope of the new Application

CHAPTER 5. SOLUTION

According to the full scope has been built in the previous chapter, the tools has been selected and the goals have been stated in the chapter 3 and chapter 1, the detailed solution states in this chapter. The solution follows the structure of the scope which divided into 3 parts which are discrete event simulation stage, data processing stage and 3D visualization stage. In this paper, the main contributions are in developing an algorithm based on the programming language Python to analyze the data from the JMT and transfer the format of data to the format accepted by the VE.bjs, and also in developing the animation functionalities and some other related functionalities in VE.bjs based on the work has done by the previous colleagues.

The structure of this chapter is that first of all, in the first section, the assumptions of the main contributions have been done in this paper have been stated. In the second to fourth chapters, there are the detailed descriptions of the work has been done in three stages and according to the main contributions have been stated before, the emphasis will be on data processing stage and 3D visualization stage.

5.1 ASSUMPTIONS

- Simulation stage: 1 buffer capacity and no failure modes of the working station
- Data processing: processing is done for one specific test case; meshes and database parameters should be defined manually.
- 3D visualization: precision of the timestamp is 0.1s; if two events which need to change position or rotation of the same mesh at the same time, there will be 200ms difference between two events; speed of conveyer and working station is constant and defined by the txt files; every time after finishing the animation, the scene should be reloaded for the next launch of the animation

5.2 DISCRETE EVENT SIMULATION STAGE

For a model definition in JMT it is necessary to describe layout of the factory in terms of queuing network model and set corresponding parameters of each buffer and server. Current test case is described by four working stations with buffer capacity equal to 1 has stated in the assumptions which have arrival rate ~ Exp(0.1) with parameters of working stations described in table below.

Table 5 Table of Parameters of Working Stations

	WS1	**WS2**	**WS3**	**WS4**
Buffer Capacity	1	1	1	1
Service Time Distribution	Det(2)	Det(2)	Det(2)	Det(4)

Queue Policy	FCFS	FCFS	FCFS	FCFS
Drop Rule	BAS Blocking	BAS Blocking	BAS Blocking	BAS Blocking

In JMT model each working station is consist of three loggers for tracking timestamp when particular entity passes through it and two service station: first one represents the buffer and the second represents the working station itself. There are two delays block in the before first working station and after fourth (last) working station representing loading and unloading. Service time distribution set to be DET(1) for both. Transportations from one working station to another is modelled using another delay block with service time distribution ~ DET(1). Finally, each buffer and working station is grouped into finite capacity region with capacity of region equal to 1. This done in order to control maximum capacity of logger and service station. Loggers are set to write all the logs in one (CSV format) file. This done to ease the processing work on the next stage.

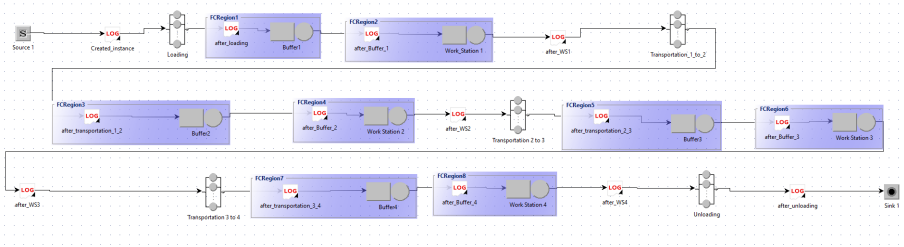


Figure 10 Model has been built in JMT.

5.3 DATA PROCESSING STAGE

The goal of this stage is to transform data from a simulation stage (input data is coming as logs) to the structure that can be used later in the visualization stage. Data processing consists of six stages: 1)action and constants definition, 2)data loading and sorting, 3)mapping entities, 4)data preparation, 5)creation structure of animation events and finally 6)data serialization. The output from data processing stage is a JSON file which describes all events (with corresponding timestamps) that later will be animated during visualization stage. All stages integrated into main function in order to be modular and comfortable for future usage.

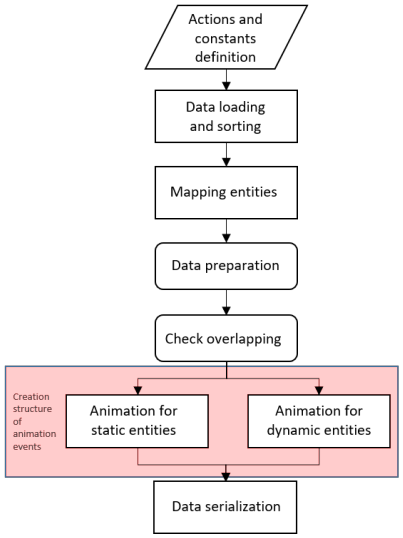


Figure 11 Flowline of Data Processing

By following the flow diagram shows above, the functions have been built in the script for each stage which are 1)"load_and_sort_logs()" for the data loading and sorting stage; 2)"mapping_entities()" for the mapping entities stage; 3) "float_to_str_rounded()" for the data preparation stage; 4) "create_animation()" and "create_ws_animation()" for the creation structure of animation events stage; 5) "animiation_to_json()" for the data serialization stage.

```
def main():
    sorted_data = load_and_sort_logs('global.csv', TIMESTAMPS_AMOUNT, NUMBER_OF_MESHES_TO_ANIMATE)
    meshes_processed = mapping_entities(sorted_data, NUMBER_OF_MESHES)
    rounded = float_to_str_rounded(meshes_processed)
    animation_file = create_animation(entity_meshes, rounded, all_events, all_infos)
    animation_ws = create_ws_animation(ws_mashes, rounded, ws_events, ws_info)
    animation_file['nodes'].extend(animation_ws)
    json_animation_file = animation_to_json(animation_file, 'Animation.json')


main()
```

Figure 12 The "main()" Function which is the main Structure of the Script

5.3.1 Action and constants definition

According to the data from a real manufacturing system, it is necessary to define precedence of actions for the specific entities (working stations, pallets, etc.) which will be visualize during the next stage. Actions are defined in terms of two lists: events list and info lists. The first one contains all the names of events that should be triggered and the second one contains additional information for some kind of events if it necessary. Also, some constants should be defined such as maximum number of moving entities in the scene, number of entities to animate (because logs file can provide enormously large number of entities) and number of logs per each job Id to correctly parse the data. Finally, all links to meshes in databases should be defined.

Table 6 Table of pre-defined Constants

Constant Name	Value	Description
TIMESTAMPS_AMOUNT	14	Number of timestamps per one job id (= number of loggers in the simulation model)
NUMBER_OF_MESHES	5	Maximum number of meshes in the scene

NUMBER_OF_MESHES_TO_ANIMATE	10	Number of job id to animate in the scene
NUMBER_OF_WS	4	Number of working station in the scene
MESHES_IN_WS	6	Number of meshes in working station assemble

Table 7 Links to Meshes in Database

Entity Type	Corresponded Mesh(Group)	Links
Pallet	Pallet.1	http://www.ontoeng.com/TestCase09#Pallet.1
	Pallet.2	http://www.ontoeng.com/TestCase09#Pallet.2
	Pallet.3	http://www.ontoeng.com/TestCase09#Pallet.3
	Pallet.4	http://www.ontoeng.com/TestCase09#Pallet.4
	Pallet.5	http://www.ontoeng.com/TestCase09#Pallet.5
Working Stations	WS01	http://www.ontoeng.com/TestCase09#WS01, http://www.ontoeng.com/TestCase09#WS01.PPs_FloatingX, http://www.ontoeng.com/TestCase09#WS01.PPs_FloatingY, http://www.ontoeng.com/TestCase09#WS01.PPs_FloatingZ, http://www.ontoeng.com/TestCase09#WS01.PPs_Gripper1, http://www.ontoeng.com/TestCase09#WS01.PPs_Gripper2
	WS02	http://www.ontoeng.com/TestCase09#WS02, http://www.ontoeng.com/TestCase09#WS02.PPs_FloatingX, http://www.ontoeng.com/TestCase09#WS02.PPs_FloatingY, http://www.ontoeng.com/TestCase09#WS02.PPs_FloatingZ, http://www.ontoeng.com/TestCase09#WS02.PPs_Gripper1, http://www.ontoeng.com/TestCase09#WS02.PPs_Gripper2
	WS03	http://www.ontoeng.com/TestCase09#WS03, http://www.ontoeng.com/TestCase09#WS03.PPs_FloatingX, http://www.ontoeng.com/TestCase09#WS03.PPs_FloatingY, http://www.ontoeng.com/TestCase09#WS03.PPs_FloatingZ, http://www.ontoeng.com/TestCase09#WS03.PPs_Gripper1, http://www.ontoeng.com/TestCase09#WS03.PPs_Gripper2
	WS04	http://www.ontoeng.com/TestCase09#WS04, http://www.ontoeng.com/TestCase09#WS04.PPs_FloatingX, http://www.ontoeng.com/TestCase09#WS04.PPs_FloatingY, http://www.ontoeng.com/TestCase09#WS04.PPs_FloatingZ, http://www.ontoeng.com/TestCase09#WS04.PPs_Gripper1, http://www.ontoeng.com/TestCase09#WS04.PPs_Gripper2

```
all_events = [['state', 'show', 'animationAdditive'], [None], [None],  ['animationAdditive'],
              [None], [None],  ['animationAdditive'], [None], [None],  ['animationAdditive'],
              [None], ['state'], ['animationAdditive'], ['hide']]


all_infos = [[{'descr': 'empty_pallet'},
              {'position': [0, 23, 100], 'rotation': [0,  0, 0]},
              {'data': 'animations/palletAdvanceToWS1.txt'}],
             [None],
             [None],
             [{'data': 'animations/palletAdvance100.txt'}],
             [None],
             [None],
             [{'data': 'animations/palletAdvance100.txt'}],
             [None],
             [None],
             [{'data': 'animations/palletAdvance100.txt'}],
             [None],
             [{'descr': 'loaded_pallet'}],
             [{'data': 'animations/palletAdvanceToEnd.txt'}],
             [None]]
```

Figure 13 Pre-defined Events and additional Information for Pallets

```
ws_events = [['state', 'state'], ['animation'], ['animation'], ['animation'],
             ['animation'], ['animation']]

ws1_info = [[{'descr': 'busy'}, {'descr': 'idle'}],
            [{'data': 'animations/PPs_FloatingX.txt', 'placementRelTo': 'http://www.ontoeng.com/TestCase09#WS01.PPs_Static'}],
            [{'data': 'animations/PPs_FloatingY.txt', 'placementRelTo': 'http://www.ontoeng.com/TestCase09#WS01.PPs_FloatingZ'}],
            [{'data': 'animations/PPs_FloatingZ.txt', 'placementRelTo': 'http://www.ontoeng.com/TestCase09#WS01.PPs_FloatingX'}],
            [{'data': 'animations/PPs_Gripper1.txt', 'placementRelTo': 'http://www.ontoeng.com/TestCase09#WS01.PPs_FloatingY'}],
            [{'data': 'animations/PPs_Gripper2.txt', 'placementRelTo': 'http://www.ontoeng.com/TestCase09#WS01.PPs_FloatingY'}]]
```

Figure 14 Pre-defined Events and additional Information for Working Stations

5.3.2 **Data loading and sorting**

During this stage, data in loaded and parsed. Data is coming from simulation tool (JMT) in a CSV format. For our purpose it is necessary to extract only two columns: timestamp and job id. After loading it is necessary to sort the data by job id and timestamp in order to use it during the next stage.

```
def load_and_sort_logs(filename, timestamps_amount, number_of_meshes_to_animate):
    all_data = []

    with open(filename, newline='') as logs:
        reader = csv.reader(logs, delimiter=";")
        next(reader, None)
        for row in reader:
            last_log_counter = 0
            row[1] = float(row[1])
            row[2] = int(row[2])
            if 0 <= row[2] <= (number_of_meshes_to_animate-1):
                all_data.append(row)
                if row[2] == number_of_meshes_to_animate-1:
                    last_log_counter += 1
            if last_log_counter == timestamps_amount-1:
                break

    sorted_data = sorted(all_data, key=itemgetter(2, 1))
    return sorted_data
```

Figure 15 Loading and Sorting.

5.3.3 **Mapping entities**

In this stage mapping job id to mesh in the scene is occurred. For mapping process, the following schema was adopted:

$$mesh_{id} = job_{id} - N * b,$$

Where mesh id – id of the mesh in the scene, job id – id of the entity in the simulation,

N- Number of meshes to animate (constant, set to be 10),

b- Batch number, $b = mesh_{id} \% N$

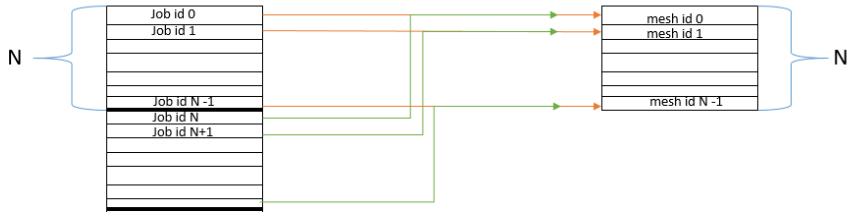


Figure 16 Logic of Mapping the Job ID

```
def mapping_entities(sorted_data, number_of_meshes):

    meshes = [[] for _ in range(NUMBER_OF_MESHES_TO_ANIMATE)]
    mapped_meshes = [[] for _ in range(number_of_meshes)]

    mesh_counter = 0

    while mesh_counter < NUMBER_OF_MESHES_TO_ANIMATE:
        j = 0
        while j < TIMESTAMPS_AMOUNT:
            meshes[mesh_counter].append(sorted_data[j+TIMESTAMPS_AMOUNT*mesh_counter][1])
            j += 1
        mesh_counter += 1

    i = 0
    batch = 0
    while i < NUMBER_OF_MESHES_TO_ANIMATE:
        if i < number_of_meshes:
            mapped_meshes[i].append(meshes[i])
        else:
            mapped_meshes[i-(number_of_meshes*batch)].append(meshes[i])
        i += 1
        if (i+1) % 5 == 0:
            batch += 1

    return mapped_meshes
```

Figure 17 Code for Mapping Entities

5.3.4 **Data preparation**

In order to prepare data for the next stage it is necessary to take into account parsing time during the next stage. Since there is a tradeoff between data precision and parsing time, precision was chosen to be 0.1s. This optional since some software simulators can support settings of precision inside simulator itself.

```
def float_to_str_rounded(logs):
    processed_logs = copy.deepcopy(logs)
    for i, mesh in enumerate(logs):
        for j, batch in enumerate(mesh):
            processed_logs[i][j] = list(map(lambda ts: str(round(round(ts, 2) * 1000, -2)), batch))

    return processed_logs
```

Figure 18 Codes for the Data Preparation Stage

5.3.5 **Creation structure of animation events**

In this stage structure of events in created. For each mesh in the scene node is created. Name of the node represents mesh name and all events inside node corresponds only to this node. Inside each node, an action list contains triggers (timestamps) and events (show, hide, animation, etc.). This information is used later for visualization entities during next stage. Creation each node is split into two stages: check overlapping entities (pallets in chosen test case) and animation for static/dynamic entities (working station/pallets).

- **Check Overlapping**

In order to avoid dynamic entities overlapping it is necessary to manage timestamps properly. Comparison of adjacent timestamps is used as a rule to find out if it is necessary to use buffer or not. Later this information is used when creating animation list for static and dynamic entities.

```
def check_overlapping(current_timestamps):
    buffers = []

    if current_timestamps[1] != current_timestamps[2]:
        buffers.append(1)
    if current_timestamps[4] != current_timestamps[5]:
        buffers.append(2)
    if current_timestamps[7] != current_timestamps[8]:
        buffers.append(3)
    if current_timestamps[10] != current_timestamps[11]:
        buffers.append(4)

    return buffers
```

Figure 19 Codes for Checking Overlapping

- **Animation for Static Entities**

During current substage nodes for static entities such as working station is created. Later list of nodes of both static and dynamic entities will merge into one list which describes behavior of the scene.

```
def create_action(timestamp, event, info=None):
    event_dict = {'trigger': {'type': 'timestamp', 'data': timestamp},
                  'event': {'type': event}}
    if info:
        event_dict['event'].update(info)

    return event_dict
```

Figure 20 Code for general Function

```
def create_node_ws(name, timestamps, events, infos):
    actions_list = []
    events_copy = copy.deepcopy(events)
    infos_copy = copy.deepcopy(infos)

    for timestamp, event, info in zip(timestamps, events_copy, infos_copy):
        actions_list.append(create_action(timestamp, event, info))

    node = {'name': name, 'actions': actions_list}
    return node


def create_ws_nodes(names, timestamps, events, infos):
    nodes = []
    ts_positions = [(2, 3), (5, 6), (8, 9), (11, 12)]
    for i in range(NUMBER_OF_WS):

        for j in range(MESHES_IN_WS):
            nodes.append(create_node_ws(names[i][j], (timestamps[ts_positions[i][0]], timestamps[ts_positions[i][1]]),
                                        events[j], infos[i][j]))

    return nodes


def create_ws_animation(ws_names, logs, ws_events, ws_infos):
    ws_animation = []
    job_id = 0
    while job_id < NUMBER_OF_MESHES_TO_ANIMATE:
        ws_animation += create_ws_nodes(ws_names, logs[job_id - (NUMBER_OF_MESHES * (job_id // NUMBER_OF_MESHES))]
        [job_id // NUMBER_OF_MESHES], ws_events, ws_infos)
        job_id += 1
    return ws_animation
```

Figure 21 Code for specific for static Entities

- **Animation for Dynamic Entities**

Animation for dynamic entities starts in the similar way as creating animation for static events. First list of events is created. The main difference is that based on the output of check overlapping function some

events can be modified and/or added. Then list of events is merged into node. Later procedure is similar as for the previous stage. Finally, list of nodes of both static and dynamic entities will merge into one list that describes behavior of the scene.

```
def create_node(name, timestamps, events, infos):
    actions_list = []
    events_copy = copy.deepcopy(events)
    infos_copy = copy.deepcopy(infos)

    for timestamp, event, info in zip(timestamps, events_copy, infos_copy):
        for sub_event, sub_info in zip(event, info):
            if sub_event:
                if ('show' in event) and ('animationAdditive' in event) and sub_event == 'animationAdditive':
                    actions_list.append(create_action(str((float(timestamp) + 200)), sub_event, sub_info))
                else:
                    actions_list.append(create_action(timestamp, sub_event, sub_info))

    node = {'name': name, 'actions': actions_list}
    return node
```

```
def create_node_with_buffers(name, timestamps, events, infos, buffers):
    actions_list = []
    events_copy = copy.deepcopy(events)
    infos_copy = copy.deepcopy(infos)

    for buffer in buffers:
        if buffer == 1:
            infos_copy[0][2]['data'] = 'animations/left_to_buffer1.txt'
            events_copy[2].insert(0, 'animationAdditive')
            infos_copy[2].insert(0, {'data': 'animations/buffer_to_nextstation.txt'})
        elif buffer == 2:
            infos_copy[3][0]['data'] = 'animations/station_to_buffer.txt'
            events_copy[5].insert(0, 'animationAdditive')
            infos_copy[5].insert(0, {'data': 'animations/buffer_to_nextstation.txt'})
        elif buffer == 3:
            infos_copy[6][0]['data'] = 'animations/station_to_buffer.txt'
            events_copy[8].insert(0, 'animationAdditive')
            infos_copy[8].insert(0, {'data': 'animations/buffer_to_nextstation.txt'})
        elif buffer == 4:
            infos_copy[9][0]['data'] = 'animations/station_to_buffer.txt'
            events_copy[11].insert(0, 'animationAdditive')
            infos_copy[11].insert(0, {'data': 'animations/buffer_to_nextstation.txt'})
```

```
    for timestamp, event, info in zip(timestamps, events_copy, infos_copy):
        i = 0
        for sub_event, sub_info in zip(event, info):

            if sub_event:
                if ('show' in event) and ('animationAdditive' in event) and sub_event == 'animationAdditive':
                    actions_list.append(create_action(str((float(timestamp) + 200)), sub_event, sub_info))
                elif len(event) == 2 and i == 1:
                    actions_list.append(create_action(str((float(timestamp) + 300)), sub_event, sub_info))
                else:
                    actions_list.append(create_action(timestamp, sub_event, sub_info))
                i += 1
    node = {'name': name, 'actions': actions_list}

    return node
```

```
def create_animation(node_names, logs, events_names, infos, trail=False):
    nodes = []
    animation = {'context': {'assetTrail': trail}, 'nodes': nodes}
    job_id = 0
    while job_id < NUMBER_OF_MESHES_TO_ANIMATE:
        buffers = check_overlapping(logs[job_id - (NUMBER_OF_MESHES * (job_id // NUMBER_OF_MESHES))]
                                    [job_id // NUMBER_OF_MESHES])
        if buffers:
            nodes.append(create_node_with_buffers(node_names[job_id - (NUMBER_OF_MESHES * (job_id // NUMBER_OF_MESHES))],
                                                  logs[job_id - (NUMBER_OF_MESHES * (job_id // NUMBER_OF_MESHES))]
                                                  [job_id // NUMBER_OF_MESHES], events_names, infos, buffers))
        else:
            nodes.append(create_node(node_names[job_id - (NUMBER_OF_MESHES * (job_id // NUMBER_OF_MESHES))],
                                     logs[job_id - (NUMBER_OF_MESHES * (job_id // NUMBER_OF_MESHES))]
                                     [job_id // NUMBER_OF_MESHES],
                                     events_names, infos))

        job_id += 1
    return animation
```

Figure 22 Code for Dynamic Events

5.3.6 **Data serialization**

The last stage in transforming format of data from dictionary (specific format used in Python) to JSON format. This stage in necessary for data being used properly during visualization stage. Later serialized data in written into file JSON format file.

```
def animation_to_json(animation, filename):
    json_file = json.dumps(animation, indent=4)
    with open(filename, 'w', encoding='utf-8') as f:
        f.write(json_file)
    return json_file
```

Figure 23 Code for Serialization

5.4 3D VISUALIZATION STAGE

The 3D visualization stage is the stage to present the simulation model in the 3D virtual environment which the scenes are more vivid and realistic. By realizing the 3D visualization stage, it is convenient for people to observe the industrial behaviors and easier for multiusers to share the information and work together.

In this paper, the main contributions in the 3D visualization stage are building the functionalities of loading and parsing the animation related data from the data processing stage and realizing the animations of the entities in the model based on the previous work has been done by other colleagues in the VE.bjs. And also, some optimizations in the Graphical User Interface(GUI) have been done to make user can operate with the application easier.

In the following sections, the functionalities for loading and parsing data have been introduced. In the second section is about the animation functionalities and the related changes in GUI and the last section is about other functionalities developed in GUI.

5.4.1 Data Loading and Parsing

The main source for the animation data is integrated in a JSON file, as what has been stated in the assumptions, which states all of the events follow the timestamp sequence. The animation JSON file is selected by the user with clicking the “Import Animation” button which has been developed in GUI.

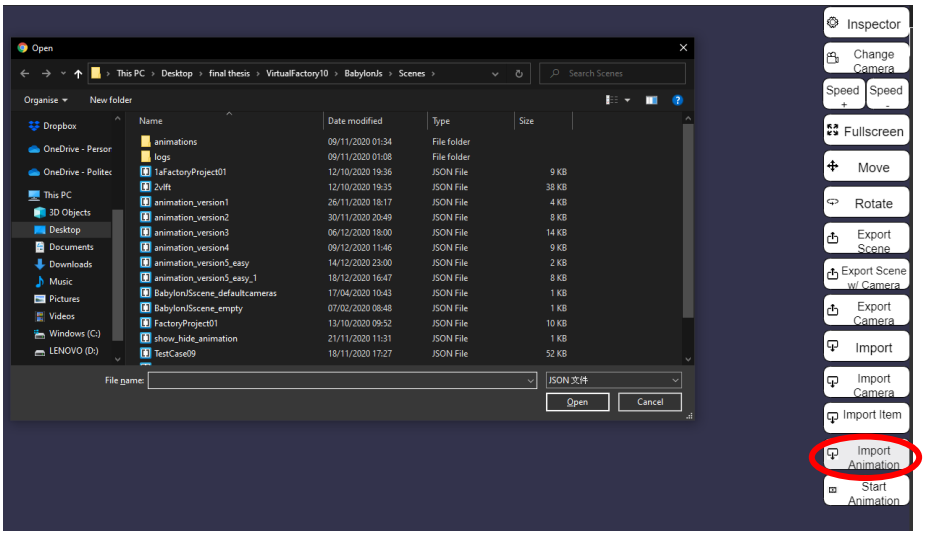


Figure 24 Importing Animation.JSON File

Meanwhile, all the information of the positions and rotations of the animation and animation additive events are saved in several local .txt files and the .txt files will be automatically found and read during the process.

The general structure of the animation json file shows in the following figure which consists of several nodes which represent different entities in the scene. For each of the node include several events which

are state, show, hide, animation, animationAdditive and trail. Each of these events represent different actions for one node during the animation.

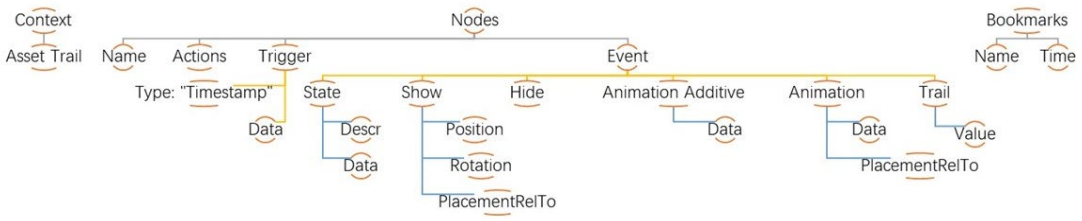


Figure 25 Structure of Animation JSON File

The state events are the descriptions of the state changes of the entities during the animation; the show and hide events let the entities do the show and hide actions; the animation and animationAdditive are the core events for the entities movements in the scene while the difference between animation and animationAdditive is that the animation event is based on the relative global coordinates and the animationAdditive event is referred to the local coordinates of the current position and rotation of the entity.

The core code for parsing the json file is called “JSON.parse()” in the function of “Import Animation” button which the system can automatically convert the local file into arrays and save the data in the system.

```
// here we tell the reader what to do when it's done reading...
reader.onload = readerEvent => {
    var content = readerEvent.target.result; // this is the content
    var filejson = JSON.parse(content);
    //here should add the function for loadJSONAnimation
    loadJSONAnimation(engine, scene, canvas, filejson)
///
}
```

Figure 26 Codes for load selected Animation.JSON File

The function “loadJSONAnimation” has been developed which is the main function to parse data. The general structure of the function shows in the following picture which follows the structure of the JSON file and create variables corresponded to each of the parameters in JSON file.

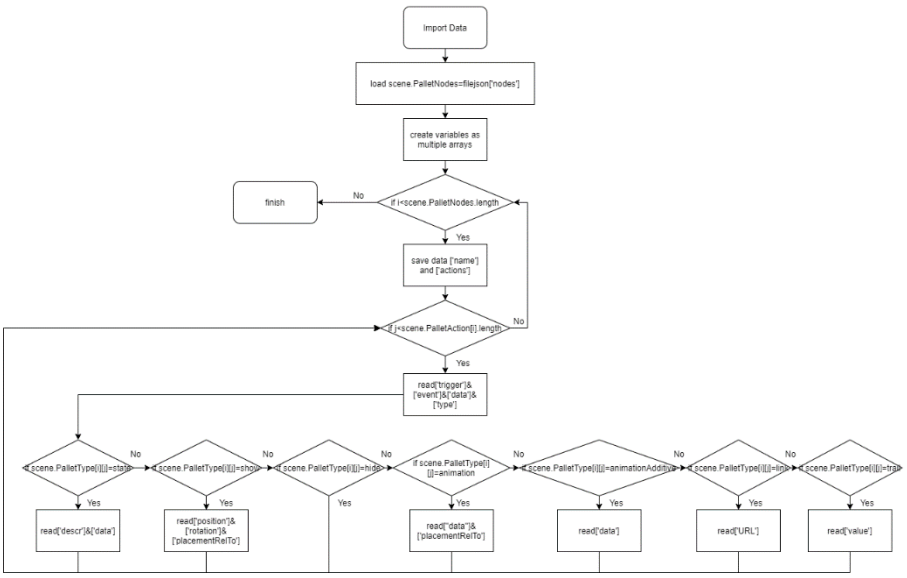


Figure 27 Logic of saving all the Parameter into the System.

The data has been parsed as the example shows in the following picture, the first dimension of the arrays which the amount is 23 are the total meshes those have events according to the json file and the second dimension of the arrays which, for example, in array 0 the length is 15 which means the total number of events for the first mesh in the JSON file is 15.

Figure 28 Parameter Structure shows in Console Log in Browser

This is the first step of parsing the JSON file which to save the parameters into the system with the same structure. For the next step is to read all the txt files according to the links in the [‘animation’] [‘data’]and [‘animation Additive’] [‘data’] in the JSON file which corresponds to the movements of the meshes during the animation and animation additive.

5.4.1.1 Related .txt File Loading and Parsing

In the animation.JSON file, which is imported into the system, the "data" of animation and animationAdditive events represent as links to the local text file as shows below.

```
{
  "trigger": {
    "type": "timestamp",
    "data": "10000"
  },
  "event": {
    "type": "animation",
    "data": "animations/palletNextStation.txt",
    "placementRelTo": "http://www.ontoeng.com/TestCase09#WS01.PPs_PalletPosition"
  }
},
```

Figure 29 Animation Event in Animation.JSON File

```
{
  "trigger": {
    "type": "timestamp",
    "data": "7000"
  },
  "event": {
    "type": "animationAdditive",
    "data": "animations/pallet90degRotation.txt"
  }
},
```

Figure 30 AnimationAdditive Event in Animation.JSON File

Those txt files contain the information of the total frames of animation, frames per seconds and position and rotation of meshes for each frame. To read these data from the txt file, the code XMLHttpRequest() has been used to call the text file from the local folder.

```
//new-built function for load txt file from the local folder
function LoadTXTFile(animation_file){
    var responsetextsave
    var allLines=[]
    var animation_text=[]
    var request = new XMLHttpRequest();
        request.open("GET", "Scenes/" + animation_file);
        console.log(request)
        request.addEventListener('load', function () {
            if (request.status >= 200 && request.status < 300) {//successfully respond
                console.log("input txt file: ",request.responseText);
                responsetextsave=request.responseText
                allLines = JSON.parse("["+responsetextsave.split(/\r\n|\n/).join(",")+"]");
                animation_text.push(allLines)
            }
        });
    request.send();
    return animation_text
}
```

Figure 31 Codes of reading local .txt Files

After the local txt file has been found, the data has been re-edited with two steps which shows in the figure before, first is to split the data with "\n" which means new line in JavaScript and then by using "join()" to add the "," into the code to replace the new line to build the data in a JSON form. Then, the JSON.parse() code can be used to parse the data into arrays to make convenience for the later operations.

During saving the information of frames, fps, positions and rotations in different variables after calling the function to find the local txt file, due to the lag of reading the local file, the code setTimeout() need to be used to let the system wait until finish loading the local text file to run the next lines of codes.

```
//save data loading from txt files
var animation_file
scene.animationdata=[]
scene.sequencesPosition=Array.from(Array(scene.AnimationNameList.length),()=>new Array())
scene.sequencesRotation=Array.from(Array(scene.AnimationNameList.length),()=>new Array())
var newline=[]
for(var i in scene.AnimationNameList){
    animation_file=scene.AnimationNameList[i]
    newline[i]=LoadTXTFile(animation_file)
}
setTimeout(function(){
    scene.sequencesID=scene.AnimationNameList
    for(var i in scene.AnimationNameList){
        scene.sequencesFrames[i]=newline[i][0][0]
        scene.sequencesFPS[i]=newline[i][0][1]
        for(var j=2; j<2+scene.sequencesFrames[i]; j++){
                scene.sequencesPosition[i].push(newline[i][0][j])
        }
    }
    for(var i in scene.AnimationNameList){
        for(var x=2+scene.sequencesFrames[i]; x<2+2*scene.sequencesFrames[i];x++){
                scene.sequencesRotation[i].push(newline[i][0][x])
        }
    }
```

Figure 32 Codes of saving Parameters from local .txt Files.

For the convenience of defining the positions and rotations when running the animation, the positions and rotations have been saved as the format of Vector3 in the system likes the code shows below.

```
position=new BABYLON.Vector3(scene.sequencesPosition[i][j][0],scene.sequencesPosition[i][j][1],scene.sequencesPosition[i][j][2])
```

Figure 33 Codes of building Position Data in format of Vector 3

5.4.1.2 Timestamp Construction

After all the data from the animation.JSON file have been read and parsed into the system, in order to avoid huge lags when the data are being read during the animation execution, the format of the data need to be reconstructed into a form has been build which is based on the timestamp sequence. To build the format with the sequence of the timestamp, the new array should be built and since in each time spot, there might be more than one mesh has events, so the arrays have been structured as 2D arrays which the codes show below.

```
scene.Eventarr=Array.from(Array(k/100+1),()=>new Array(n))
```

Figure 34 Codes for constructing Arrays.

In the codes show above, the k is the value of the maximum timestamp according to the last timestamp in animation.JSON file and in that file the unit of timestamp is milliseconds. For the sake of accuracy but also to keep having lags as small as possible, in this paper the timestamp unit has been defined as 0.1s which is synchronized as what has been defined in the data processing stage and this is the reason why in the codes above, the k value has been divided by 100.

After constructed all the related arrays, by using the for loops, all the data has been parsed and saved in the previous step which has been introduced in the previous section will be saved following the timestamp sequence which as shown in the console log in the browser in the figure below.

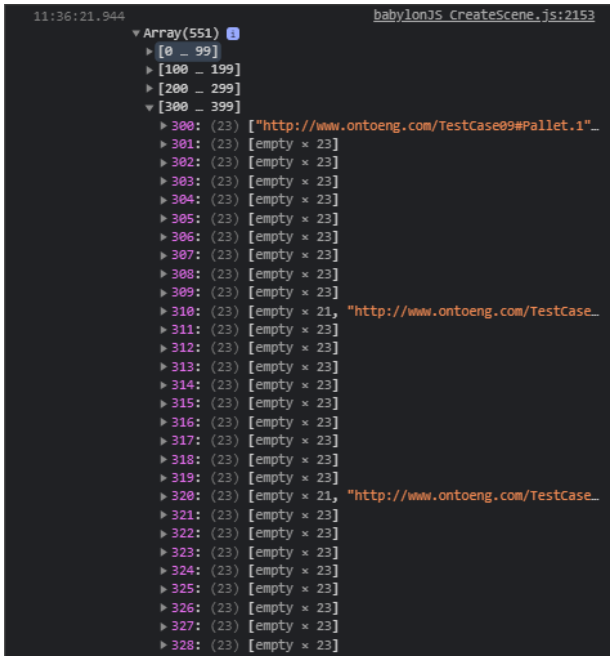


Figure 35 Console Log of the Data with the Sequence of Timestamps

5.4.2 **Animation Functionalities**

After all the data have been parsed in the format which following the sequence of the timestamp, the foundation of running the animation has been built. To make the animation present, the next step is to build the functions to let the meshes do the desired actions according to different events. Generally, there are 6 types of events which are state event, show event, hide event, animation event, animation additive event and trail event. For each of these events, a corresponding function has been built to realize either the animation movements or the functionalities in the GUI.

5.4.2.1 Developments of the Graphic User Interface of Animation Functionalities

After imported the animation json file by clicking the “Import Animation” button, to launch the animation, the button “Start Animation” should be clicked and once the button has been clicked, the panels for controlling the animation show in the scene which can be divided into two parts.

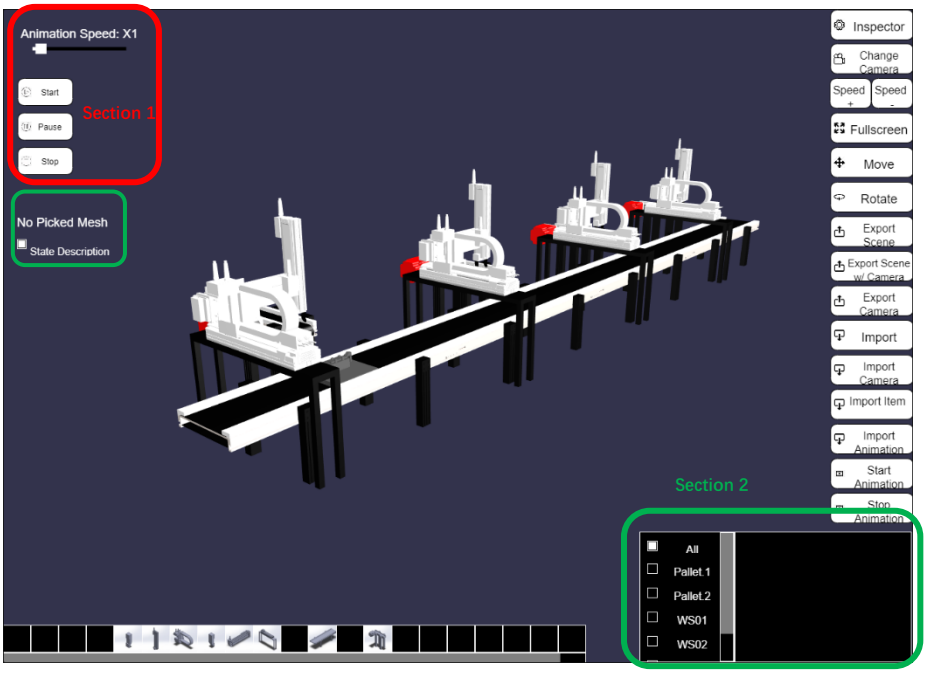


Figure 36 Start Animation Interface

In the left side in the interface, there is a slider for adjusting the animation speed from x0.1 to x10, three buttons which are “Start”, “Pause” and “Stop” for controlling the process of the animation. Below the “Stop” button is a text box which shows the name of the mesh when any mesh has been clicked on and a checkbox to let the user to choose whether to show the state descriptions or not.

On the right-bottom part, there are two scroll viewers and a new “Stop Animation” button. For the scroll viewer on the left side shows several checkboxes with the names of the meshes which are automatically generated.

The right scroll viewer shows the state of the meshes during running the animation and the data will be continuously updated along the simulation time. If there are some checkboxes has been checked in the left scroll viewer, in the right scroll viewer will show only the states of the specified meshes.

Once the animation is finish, by clicking the “Stop Animation” button, the start animation interface will be disposed, and it will return to the main interface.

```
    //Stop Animiation Button
    var button13 = createCustomButton("but13", "Stop Animation", "Assets/animation.png", advancedTexture, 700);
    button13.onPointerClickObservable.add(function () {
    slider_panel.dispose();
    timepanel.dispose();
    animationbutton1.dispose();
    animationbutton2.dispose();
    animationbutton3.dispose();
    button13.dispose();
    meshidpanel.dispose();
    svleft.dispose();
    svright.dispose();
    });
});
```

Figure 37 Codes for Stop Animation Button

- Animation Control Section

The animation control section includes three buttons and one slider which monitors the main animation process. To let the animation, speed up or slow down, start, pause or stop. This section is the main section to let the user to control the whole animation process.

1) Animation Speed

The functionality of the animation speed is to let the user to customize the running speed of the animation, it can be speed up or slow down depends on the user's needs. The slider has been used to realize this functionality, and the range of the slider is from x0.1 to x10.

Figure 38 Animation Speed Adjustment

The value of the slider will be automatically saved to be used on the running timer and the simulation time.

```
myInterval=setInterval(MyFunction,1000/slider1.value);
```

Figure 39 Value of the Slider works on the Simulation Time

2) Start Button

The start button is used to launch the animation once all the required files has been imported and the specified parameter according to the user requirements has been chosen. Once the button has been clicked the simulation of the animation will start, meanwhile in the top-middle of the screen there will appear a timer which shows the text "Simulation Start" plus the corresponding time of the simulation.

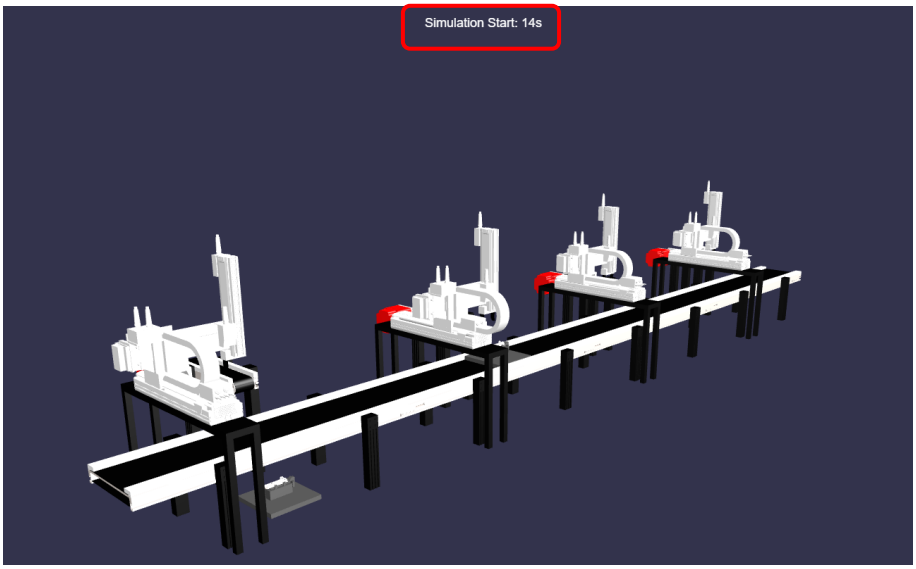


Figure 40 Simulation Time shows in the Scene.

3) Pause Button

The pause button is used to pause the animation but not stop the whole process. Once the simulation has been paused, the animation in the scene will stop and the text of simulation clock will change to “Simulation Paused” plus the corresponded time when the simulation paused. If the start button has been clicked again, the simulation will continue from the moment it has been paused not from the beginning.

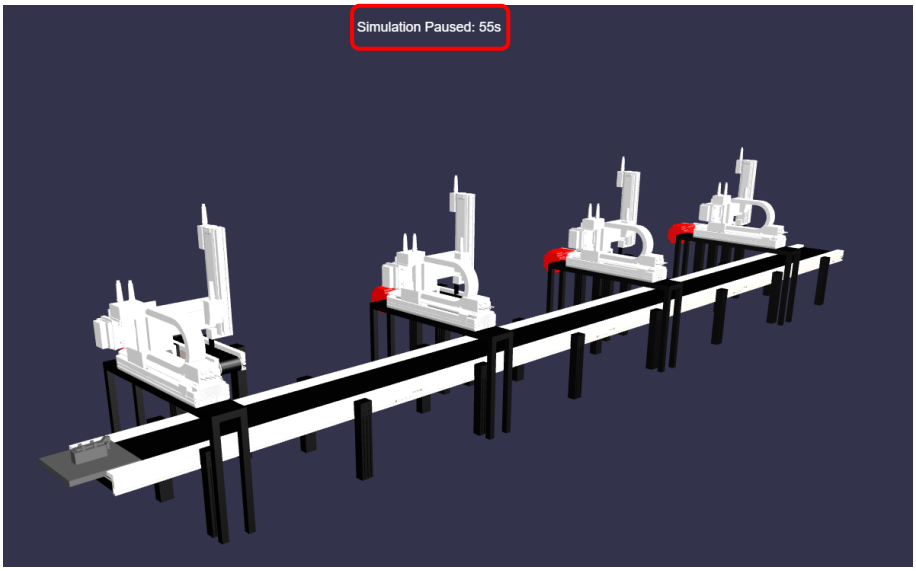


Figure 41 Simulation Clock Paused

4) Stop Button

Once the stop button has been clicked, the whole animation will stop and, in the screen, the scene of the last moment before the button has been clicked will be keep there and the simulation clock will show the text of “Simulation Stop” plus the time when the simulation stopped until the start button has been clicked. Once the start button has been clicked, the whole animation will start from the beginning.

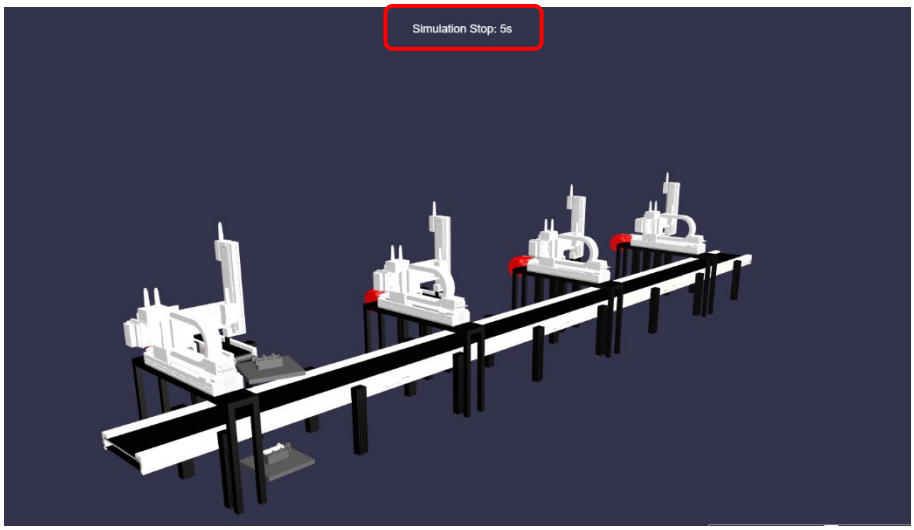


Figure 42 Simulation Clock Stopped

5.4.2.2 Animation

The animation event is to let the mesh move and rotate always based on a relative global coordinate which is different from the animation additive.

First of all, in the codes below show the format of the animation events in the animation.JSON file. In the codes, there are two key parameters which are the “data” and “placementRelTo”. For the content in “data” is a link to the local txt file which has already been loaded and parsed when import the animation file. The other parameter is “placementRelTo” which means the movements of the mesh is related to the position and rotation of the mesh in “placementRelTo”.

```
{
  "trigger": {
    "type": "timestamp",
    "data": "10000"
  },
  "event": {
    "type": "animation",
    "data": "animations/palletNextStation.txt",
    "placementRelTo": "http://www.ontoeng.com/TestCase09#WS01.PPs_PalletPosition"
  }
},
```

Figure 43 Animation Events in Animation. JSON File

Thanks to the strong library of Babylon.JS, the animation engine has already been built in Babylon.JS and a certain structure which mainly contains three steps to realize the animation functionality which are:

1) create BABYLON.Animation(),

2) create animation keys,

3) to execute the animation with the code beginDirectAnimation().

For create the BABYLON.Animation(), thanks to the Babylon.JS documentations, it has been programmed in a standard form which as shown in the following picture.

```
new Animation(name: string, targetProperty: string, framePerSecond: number, dataType: number, loopMode?: number | undefined,
enableBlending?: boolean | undefined): Animation
```

Figure 44 Definition of Animation Function in Babylon Library

According to this standard format, the animation has been created as shows in the following codes.

```
//create Animations
var PalletAnimation=new BABYLON.Animation('position animation','position',seqfps,BABYLON.Animation.ANIMATIONTYPE_VECTOR3,BABYLON.Animation.ANIMATIONLOOPMODE_CYCLE)
var PalletRotationAnimation=new BABYLON.Animation('rotation animation','rotation',seqfps,BABYLON.Animation.ANIMATIONTYPE_VECTOR3,BABYLON.Animation.ANIMATIONLOOPMODE_CYCLE)
```

Figure 45 Codes of creating Animations.

In the brackets of the BABYLON.Animation, the first parameter is the name of this animation. The second parameter is the target property which in the first line of codes is the displacement and in the second is the rotation. The third parameter is the frame per second(fps) of the animation. The fourth parameter is the data type, since all the meshes in the scene have been built in BABYLON.VECTOR3,

so at here, the data type is VECTOR3 as well. The last parameter is the loop mode, since the animation in the scene will be continuously executed, so at here the loop mode is cycle.

After the Animation has been built, the next step is to set the animation keys which is the main content when the animation is running. Before setting the keys, because of the “placementRelTo” which has been mentioned above, an if statement has been used to judge whether the position and rotation value which will be saved into keys are absolute or relative. The solution is that if in the data there is “placementRelTo”, the position and rotation of the related mesh will be saved into the parameter “related_mesh_position” and “related_mesh_rotation”, if there is no “placementRelTo”, these two parameters will equal to 0. The parameters “related_mesh_position” and “related_mesh_rotation” will be the initial value when launch the animation.

```
//find relative mesh
if(temp_mesh.parentObject==undefined){
    var related_mesh_position=related_mesh.position.divide(related_mesh.scaling)
    console.log(related_mesh_position)
    var related_mesh_rotation=related_mesh.rotation
}else{
    var related_mesh_position=new BABYLON.Vector3(0,0,0)
    var related_mesh_rotation=new BABYLON.Vector3(0,0,0)
}
```

Figure 46 Codes for realizing "placementRelTo"

After done with the parameter “placementRelTo”, the keys can be built in arrays, by using for loop to save all the position information and rotation information into the keys. Which shows in the codes below. What need to pay attention is that since the meshes which are imported into the scene, their original size unit are different from each other, so to make sure all the meshes are in the same unit, the scaling has been used to adjust the size of each mesh. In this case, when the position and rotation of the meshes which are imported here should also consider the scaling since to make sure all the measuring units are the same. As shown below, before the for loop, a parameter has been built to find the original scaling of the mesh and calculate the actual position and rotation with division of this parameter.

```
//set Animation Keys
var keys=[];
var rotationkeys=[];
//console.log(scene.MySequencePosition);
var temp_mesh_scale=new BABYLON.Vector3(temp_mesh.scalingOrig,temp_mesh.scalingOrig,temp_mesh.scalingOrig);
console.log(temp_mesh_scale)

//position keys
for(var z=0; z<seqframe;z++){
    var temp_mesh_value=scene.MySequencePosition[seqid][z].divide(temp_mesh_scale)
    var temp_mesh_value=temp_mesh_value.multiply(temp_mesh.scaling)
    keys.push({
        frame:z,
        value:related_mesh_position.add(temp_mesh_value),//babylon vector_3
    })
}
console.log(keys)

//rotation keys
for(var z=0; z<seqframe;z++){
    rotationkeys.push({
        frame:z,
        value:related_mesh_rotation.add(scene.MySequenceRotation[seqid][z]),//babylon vector_3
    })
}
console.log(rotationkeys)
```

Figure 47 Codes of setting Animation Keys

Since the animation has been built and the animation keys has been created, before executing the animation, the keys should be saved into the animation which by using the codes shows below.

```
PalletRotationAnimation.setKeys(rotationkeys);
PalletAnimation.setKeys(keys);
```

Figure 48 Codes for relating Keys with Animation.

The final step is to execute the animation with the code beginDirectAnimation which has been predefined by Babylon.JS.

```
//execute the animation
var anim=scene.beginDirectAnimation(temp_mesh,[PalletAnimation,PalletRotationAnimation],0,seqframe,false)
```

Figure 49 Codes for executing the Animation.

Inside the brackets of the beginDirectAnimation, the first parameter is the target mesh, the second matrix includes all the animation to be executed, the third and fourth parameters are the start and the end of the executing frames and the last Boolean value equal to false means the animation won't repeat.

5.4.2.3 Animation Additive

The biggest difference between animation and animation additive is animation is based on the coordinates of the initial position and rotation of the mesh, but for the animation additive, each movement is based on the current coordinates which means with the movement of the mesh, the reference coordinates are always changing.

To realize the animation additive, the solution has two steps. The first is to modify the input data from the txt file. Since animation additive in position and rotation changes is the accumulation of the input data, so for example the txt file shows on the left side below, when the data is being reading and parsing into the system, it can make a summation as shows on the right side below. So, by using ".add()" function, the summation of the previous position and the current position achieves.

```
10
10
[0,0,-10]
[0,0,-10]
[0,0,-10]
[0,0,-10]
[0,0,-10]
[0,0,-10]
[0,0,-10]
[0,0,-10]
[0,0,-10]
[0.0, 0.0, 0.0]
[0.0, 0.0, 0.0]
[0.0, 0.0, 0.0]
[0.0, 0.0, 0.0]
[0.0, 0.0, 0.0]
[0.0, 0.0, 0.0]
[0.0, 0.0, 0.0]
[0.0, 0.0, 0.0]
[0.0, 0.0, 0.0]
```

(a)

```
10
10
[0,0,-10]
[0,0,-20]
[0,0,-30]
[0,0,-40]
[0,0,-50]
[0,0,-60]
[0,0,-70]
[0,0,-80]
[0,0,-90]
[0,0,-100]
[0.0, 0.0, 0.0]
[0.0, 0.0, 0.0]
[0.0, 0.0, 0.0]
[0.0, 0.0, 0.0]
[0.0, 0.0, 0.0]
[0.0, 0.0, 0.0]
[0.0, 0.0, 0.0]
[0.0, 0.0, 0.0]
[0.0, 0.0, 0.0]
```

(b)

Figure 50 (a) .txt File for AnimationAdditive Events (b) .txt File for Animation Events

The next step is the same as in the animation. To find the current position and rotation of the target mesh and use them as the origin of the reference coordinate and make summations during setting keys array by array from the modified data as the codes shown below.

```
//position keys
for(var z=0; z<seqframe;z++){
    additive_keys.push({
        frame:z,
        value:itsposition.add(scene.MysequencesPositionAdditive[seqid][z]),//babylon vector_3
    })
    itsposition=itsposition.add(scene.MysequencesPositionAdditive[seqid][z])
}
console.log(additive_keys)

//rotation keys
for(var z=0; z<seqframe;z++){
    additive_rotationkeys.push({
        frame:z,
        value:itsrotation.add(scene.MysequencesRotationAdditive[seqid][z]),//babylon vector_3
    })
}
console.log(additive_rotationkeys)
```

Figure 51 Codes for setting Keys for AnimationAdditive Events

5.4.2.4 Show and Hide

The functionalities for show and hide events are mainly about to let the mesh show and hide during the animation.

The core code of this functionality is mesh.isVisible=true/false, when the Boolean value is equal to true, the mesh shows and when the value equal to false, the mesh hides. The show function and hide function have been built as shown in the following codes.

```
********Show Function************/
unction show(mesh_name){
   var temp_mesh_name =scene.getMeshByID(mesh_name);
   temp_mesh_name.isVisible =true;
   return;


********Hide Function************/
unction hide(mesh_name){
   var temp_mesh_name =scene.getMeshByID(mesh_name);
   temp_mesh_name.isVisible=false;
   return;
```

Figure 52 Show and Hide Functions

In the animation. JSON file, the format of show and hide events are like the codes show below.

```
{
  "trigger": {
    "type": "timestamp",
    "data": "2000"
  },
  "event": {
    "type": "hide"
    }
},
{
  "trigger": {
    "type": "timestamp",
    "data": "3000"
  },
  "event": {
    "type": "show",
    "placementRelTo": "http://www.ontoeng.com/TestCase09#WS01.PPs_PalletPosition",
    "position": [0,10,0],
    "rotation": [3.1415927,-0.3097395,3.1415927]
  }
},
```

Figure 53 Hide and Show Events in Animation.JSON File

As the codes show above, there are 3 parameters in “show” events which are position, rotation and placementRelTo. For the position and rotation, they are the parameter to define where the mesh is when it shows. For the placementRelTo means the position and the rotation of the mesh is relative to the mesh in the parameter placementRelTo.

As shown in the following codes, before the function “show(mesh_name)” which has been built, there is an if statement to judge whether there is the parameter “placementRelTo”. If has, the position and rotation of the mesh when it shows will be the coordinates will be the relative position and rotation to the local coordinates of the related mesh. Meanwhile, if there is no data for “placementRelTo”, it means the coordinates show in the “position” and “rotation” are the absolute value refer to the global coordinates.

```
var showpallet=scene.getMeshByID(mypallet)//target mesh
var relpallet=scene.getMeshByID(scene.ShowRelPosarr[i][j])//related pallet
if(relpallet!==null){//if has placementRelTo
    var relPos=relpallet.position
    var relRot=relpallet.rotation
    //console.log('PlacementRelTo: '+relpallet+relPos+relRot)
    if(scene.Positionarr[i][j]){
        console.log('Position of showing mesh: '+scene.Positionarr[i][j])
        var itsPosition = new BABYLON.Vector3(scene.Positionarr[i][j][0],scene.Positionarr[i][j][1],scene.Positionarr[i][j][2]).add(relPos.divide(relpallet.scaling));
        showpallet.position=itsPosition
    }
    if(scene.Rotationarr[i][j]){
        var itsRotation=new BABYLON.Vector3(scene.Rotationarr[i][j][0],scene.Rotationarr[i][j][1],scene.Rotationarr[i][j][2]).add(relRot)
        showpallet.rotation=itsRotation
    }
}else{
    if(scene.Positionarr[i][j]){
        var itsPosition = new BABYLON.Vector3(scene.Positionarr[i][j][0],scene.Positionarr[i][j][1],scene.Positionarr[i][j][2]);
        console.log(itsPosition)
        showpallet.position=itsPosition
    }
    if(scene.Rotationarr[i][j]){
        var itsRotation = new BABYLON.Vector3(scene.Rotationarr[i][j][0],scene.Rotationarr[i][j][1],scene.Rotationarr[i][j][2]);
        showpallet.rotation=itsRotation
    }
}
show(mypallet);
```

Figure 54 Codes of "placementRelTo" in Show Events

For the hide event, since there are no more other parameters, so only to call the function hide() when the event is “hide”.

5.4.2.5 State Event

There are 2 functionalities for the state events, the first one is to let the mesh change the model during the animation when it is necessary to represent some certain condition such as to represent the pallet is empty or it is onload with workpieces.

Another functionality is showing the state of the pallets and the machines during the animation which can let the users always get the information of, for example, whether the machine is busy or idle among the whole animation process.

1)Functionality to change Mesh Model.

For the functionality to change the mesh model during the animation, first of all, the information from the animation.JSON file should be parsed which contains the local file path of the model as the format shows below.

```
"event": {
  "type": "state",
  "data": "../repository/virtualLearningFactory/models/Pallet/Pallet_Wing.OBJ",
  "descr": "part loaded"
}
```

Figure 55 Information of changing Model in State Event in Animation.JSON

During the information is being parsed and saved into the system, according to the file path in 'data', the models will be loaded and imported into the scene. To find and import the model from the local file, first a new mesh should be built by using BABYLON.Mesh() and define the file path of this mesh according to the information from the JSON file. By using the predefined function import_OBJ2(), the model can be imported into the scene which shows as below.

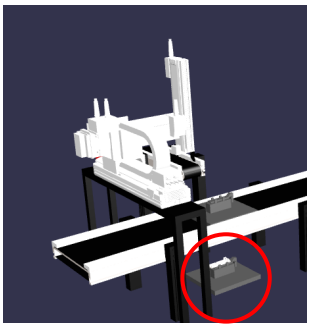


Figure 56 Imported Mesh (red circle)

2)Functionality to show the Description of Mesh State

The "mesh states" functionality is to let the user can easily check the state changes of machines and pallets during the animation. To make it more customized for the user to use, a scroll viewer has been created which is the section contains several checkboxes represent different meshes in the model can be selected to let the user only observe the information they are interested in. The number of the checkboxes are automatically generated by the input file and above all of these checkboxes there is a checkbox represents all the mesh has been selected.

For the state events, the functions can be divided into two steps which are creating the scroll viewers and checkboxes before executing the animation, and the second step is to show the information of selected meshes during the animation.

First of all, in GUI, the section for description of mesh state has been built as shows in the following figure.

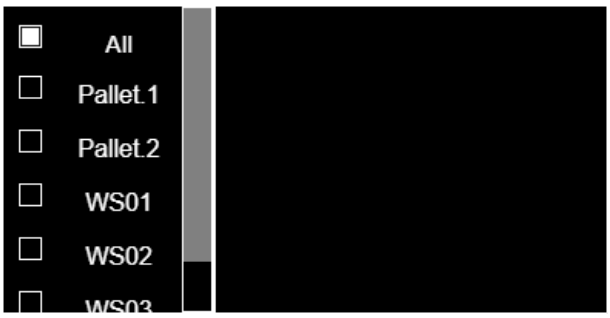


Figure 57 Mesh State Section

The initial state of the left scroll viewer is as shown in the picture above which the "All" checkbox is checked. When "All" is checked, in the right scroll viewer the information of all meshes will display. Once the user selects one of the following checkboxes, the value of the first "All" checkbox will automatically change to false which means the checkbox is unchecked. Meanwhile, during the animation process, the right scroll viewer will only show the information of the mesh which has been selected in the left scroll viewer.

(a) (b)

Figure 58 (a) All Meshes selected (b) Specific Meshes checked

For the first step, two scroll viewers have been built at the right bottom corner in the scene which for the left side one contains several checkboxes which is depends on the number of the meshes those have the state events, and the right side scroll viewer is the main panel to show the state of the meshes during the animation process. The details about how it looks in the scene has already been introduced in the section 4.2.2.2. To realize those structures in the scene, here there are four functions from the Babylon.JS have been mainly used which are BABYLON.GUI.StackPanel(), BABYLON.GUI.ScrollViewer(), BABYLON.GUI.Checkbox() and BABYLON.GUI.TextBlock(). The hierarchy of them is first to build the StackPanel and add ScrollViewer to the StackPanel. Then,

create a new StackPanel which is added to ScrollViewer and add CheckBoxes and TextBlockes to this StackPanel at the end. The structure show in the following figure.

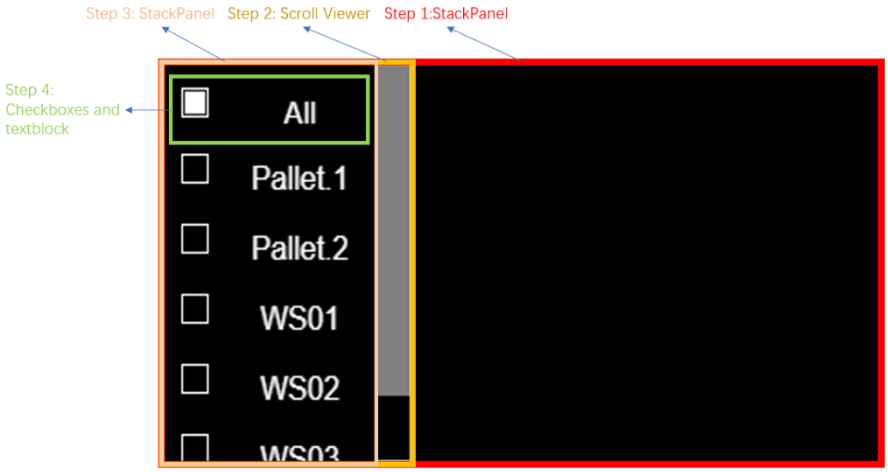


Figure 59 Structure of State Description Section

After the structure has been built, all the checkboxes need to define the functions to realize the functionality that when the "All" has been checked, other checkboxes change to unchecked state and the information of all meshes shows in the right scroll viewer when launching the animation. While when some of the checkboxes are checked, the checkbox "All" changes to unchecked state and during the animation only the information of the selected meshes will be shown. In the following codes shows how to realize these functionalities in codes.

```
//Functions for Checkboxes
for(var i in checkboxname){
    checkboxname[i].onIsCheckedChangedObservable.add(function(value) {
        if(value===true){
            allchecked.isChecked=false
        }
    })
}

allchecked.onIsCheckedChangedObservable.add(function(value){
    if(value===true){
        matrix_for_saving_description=scene.PalletState
        console.log(matrix_for_saving_description)
        for(var i in checkboxname){
            checkboxname[i].isChecked=false
        }
    }else{
        matrix_for_saving_description=[]
        console.log(matrix_for_saving_description)
    }
})
```

Figure 60 Functions for Checkboxes of State Events

In the codes show above, the first for loop is to define that when some of the specific meshes' checkboxes are checked, the "All" checkbox will turn to unchecked. And for the second paragraph, it defines that when the "All" checkbox is checked, all of other checkboxes are unchecked.

In addition, another functionality for selecting the mesh to show its information during the animation is to directly click on the mesh which will also automatically make the corresponded checkbox to be checked. The following codes show below are to realize this functionality.

```
var meshforstate=[];
const listenTo = id => {
    const canvas = document.getElementById(id);
    canvas.addEventListener('click', () => console.log(`${id} : click`));
}
canvas.addEventListener('click', () => {
    const result = scene.pick(scene.pointerX, scene.pointerY);
    var pickedMesh;
    var pickedmeshstate;
    if (result.hit) {
        if(result.pickedMesh.parentObject!==undefined){
            pickedMesh = result.pickedMesh.parentObject;
            console.log(pickedMesh);
            meshid.text= getLocalUri(pickedMesh);
            pickedmeshstate=pickedMesh;
        }else{
            pickedMesh = result.pickedMesh;
            meshid.text= getLocalUri(pickedMesh.id);
            pickedmeshstate=pickedMesh.id;
        }
    }
    if(pickedMesh !==undefined){
        if(meshforstate.indexOf(getLocalUri(pickedmeshstate))==-1){
            meshforstate.push(meshid.text);
            for(var i=0; i<k;i++){
                if(meshforstate.indexOf(checkboxname[i].id)!=-1){
                    checkboxname[i].isChecked=true;
                }
            }
        }
    }
})
```

Figure 61 Codes for selecting Meshes by clicking on the Mesh.

The next step is to show the information of the meshes during the animation process. To realize the functionality which can find which checkbox is checked and the condition of it, at the beginning of running the animation, there is a section to find all the checked checkboxes and save the id of them which can be used in the later functions for state events to find the corresponding description at a certain timestamp. The following codes realizes these functionalities.

```
//Save Checked Meshes
var checkedmeshposition=[];
for(var x=0;x<k;x++){
    if (checkboxname[x].isChecked===true){
        checkedmeshposition.push(x);
        matrix_for_saving_description.push(scene.PalletState[x]);
    }
}
```

Figure 62 Codes for finding and saving checked Checkboxes.

Then, during the animation, the state event is divided into two part which are if the "All" checkbox is checked and if some specific checkboxes are selected.

```
//STATE EVENT
if(scene.Typearr[i][j][z]=='state'){
    if(matrix_for_saving_description!=[]){
        if(matrix_for_saving_description.indexOf(mypallet)!=-1){
            var stateExam=scene.StateDescrarr[i][j]
            var stateName=getLocalUri(mypallet)
            statetext=statetext+'(At '+i+'s)'+stateName+': '+stateExam+'\n'
            stateid.text=statetext
        }
    }else if(checkedmeshposition!=[]){
        for(var x=0;x<checkedmeshposition.length;x++){
            console.log(checkboxname[x])
            if(checkboxname[x].id.indexOf(mypallet)!=-1){
                var stateExam=scene.StateDescrarr[i][j]
                var stateName=getLocalUri(mypallet)
                statetext=statetext+'(At '+i+'s)'+stateName+': '+stateExam+'\n'
                stateid.text=statetext
            }
        }
    }
```

Figure 63 Codes for executing the State Events.

So, in the codes above is the codes for the state event which has an if judgment to realize this. For the first part of the if judgement is when there are some specific checkboxes are checked, the codes are trying to find the state of the mesh inside the data which has been parsed and saved in the system according to the id of the checkboxes and then states the events which they have at a certain timestamp and the text will be like the example shows below.

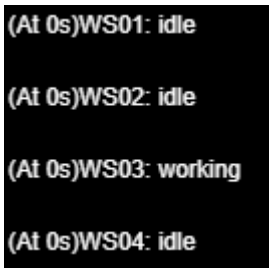


Figure 64 Description of States during the Animation

5.4.2.6 Trails

The trail event is the event to let the mesh has a track which can make the user easier to realize the movement of the mesh. In the animation.JSON file, the format of trail event shows below.

```
{
  "trigger": {
    "type": "timestamp",
    "data": "10000"
  },
  "event": {
    "type": "trail",
    "value": true
  }
},
```

```
{
  "trigger": {
    "type": "timestamp",
    "data": "25000"
  },
  "event": {
    "type": "trail",
    "value": false
  }
} ,
```

(a) (b)

Figure 65 Trail Events in Animation.JSON File (a) Turn on the Trail (b) Turn off the Trail

The parameter of the trail event is a Boolean which only has the value true and false. When the value equal to true, the trail appears, when it is equal to false, the trail will disappear.

Thanks to the library of Babylon.JS, it has its own code for build the trail which is BABYLON.TrailMesh(name, generator, scene, diameter, length, autoStart). So the trail has been built as shows in the following code,

```
var newTrail = new BABYLON.TrailMesh('pallet_trail',mytrail,scene,10,100)
```

Figure 66 Codes for creating the Trail in Babylon

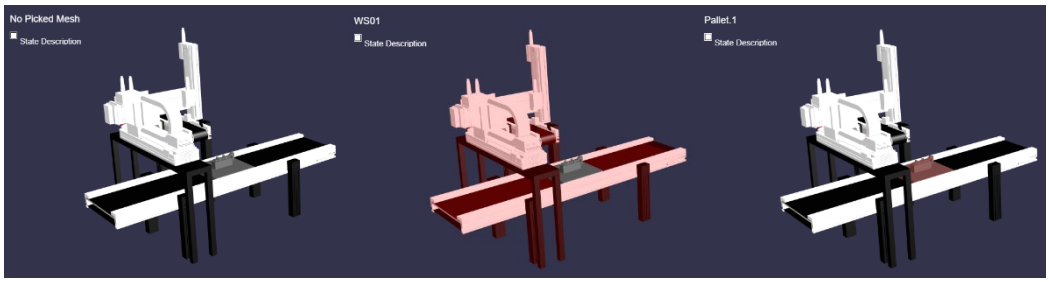
Since in the JSON file, there are Boolean value to control the turn on and turn off of the trail, so an if statement has been used to define the switch of the trail which shows in the following codes.

```
//TRAIL EVENT
}else if(scene.Typearr[i][j][z]=='trail'){
    if(scene.TrailValuearr[i][j]=true){
        var mytrail=scene.getMeshByID(mypallet)
        var newTrail = new BABYLON.TrailMesh('pallet_trail',mytrail,scene,10,100)
        var sourceMat=new BABYLON.StandardMaterial('sourceMat',scene);
        var color=new BABYLON.Color3.White()
        sourceMat.diffuseColor = color;
        newTrail.material = sourceMat;
    }else if(scene.TrailValuearr[i][j]=false){
        newTrail.stop();
    };
};
```

Figure 67 Codes for turning on and off the Trail

5.4.3 Mesh Picking Functionalities in Graphical User Interface

There is another functionality has been built in GUI for the aim of making convenience for users which is mesh picking. The mesh picking is that when the user clicks a random mesh or the mesh group in the scene, the name of the mesh or the mesh group will be shown in the left-middle side of the screen as in the following picture. When there is no mesh has been picked, it shows in the textbox “No Picked Mesh”.



(a) (b) (c)

Figure 68 (a) No Mesh picked (b) Workstation is picked (c) Pallet is picked

There are two purposes for building the mesh picking functionality. The first purpose is to let the user be able to easily know which mesh they are clicking. it makes more convenient for the user to understand the scene and also easier to find their desired mesh.

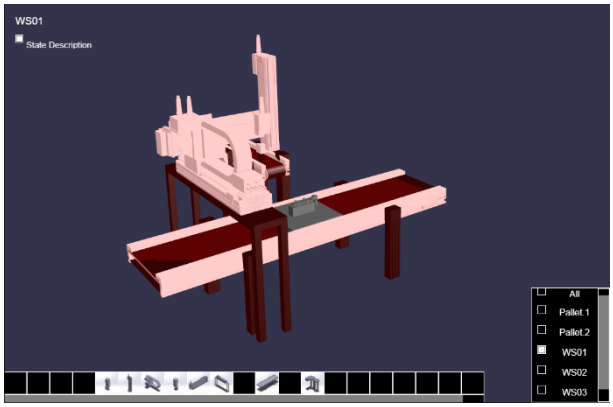


Figure 69 Corresponding Checkbox is checked when the Mesh is picked.

The second purpose is to link with showing the state of mesh in the scroller viewer in the right bottom. Once the user has clicked any of the mesh, the corresponding checkbox in the left scroll viewer will be automatically checked, and during the simulation, the right scroll viewer will only show the state's detail of that picked mesh.

CHAPTER 6. EXPERIMENTS AND VALIDATION

During the validation stage full cycle of data transformation starting from defining simulation model using JMT, following by data processing in Python and finally running visualization in Javascript using BabylonJs was tested. For this purpose, test case for the simulation and data processing stages was created (see 5.2, 5.3). All new functionality in visualization stage was tested.

6.1 MAIN FUNCTIONALITY CHECK

- GUI

For the GUI development, after imported the test case, the scene has been loaded and at the right bottom corner, the checkboxes of the information of the entities in the scene has been automatically generated. Also, in the scene, the mesh in the red circle which is loaded from the local file has been imported into the system.

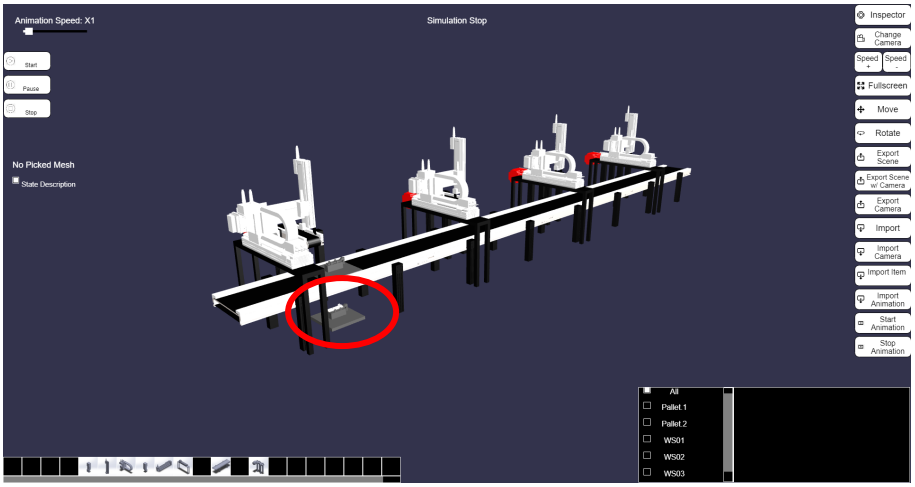


Figure 70 Test of the GUI Functionalities

- Mesh State Check

For the mesh state check functionality, as shown in the figure below, when the entity in the scene has been selected (in this case is working station 1), the information about the name of the entity shows in the left middle of the scene meanwhile in the right bottom part of the scene about the state description, the corresponded checkbox of the WS01 has been automatically checked which means during the animation, only the description of the state changes of the WS01 will show.

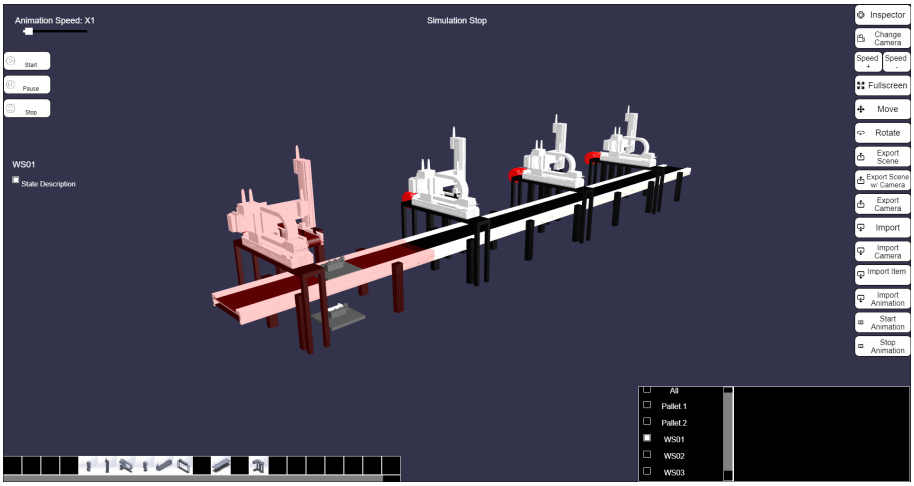


Figure 71 Test of Mesh State Functionality

- Running the Animation

As shown in the figures below, the animation in the virtual environment works according to the goals expected. The pallets and the working stations in the scene move correctly and in the section of the descriptions of the entities state keeps updating along the animation time.

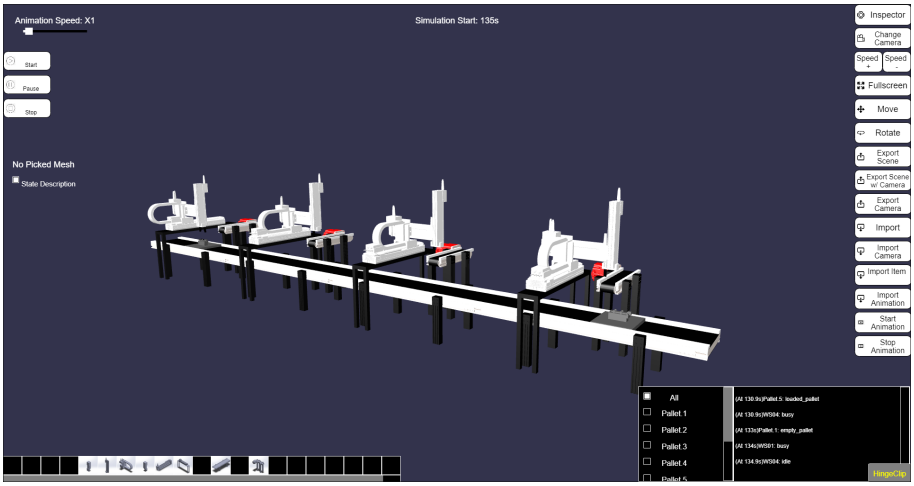


Figure 72 Screenshot of the Animation

When all the animation finish, a window pops up with the text “the animation is finished” and the scene stops at the last moment of the animation.

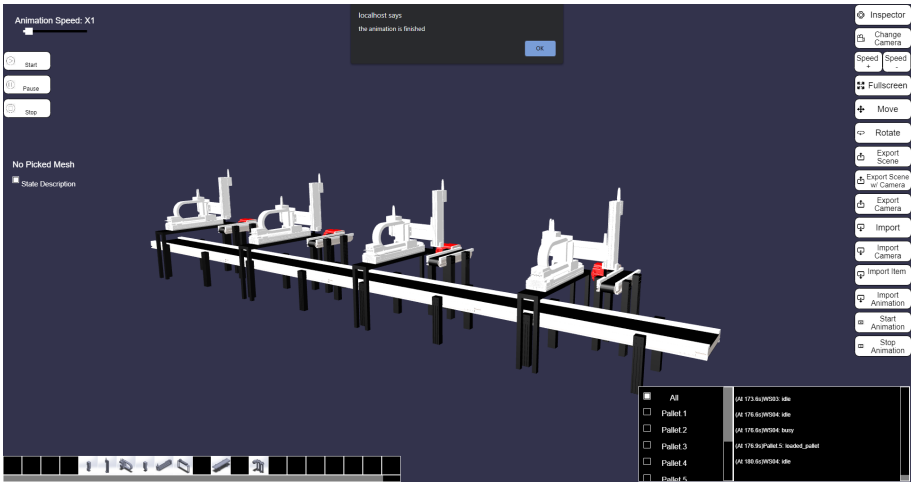


Figure 73 The last Moment when the Animation is finished.

- Buffer

  As it can be seen from the figure, entity (pallet) moves to buffer instead of working station when working station in busy. And when it's idle entity moves directly to working station

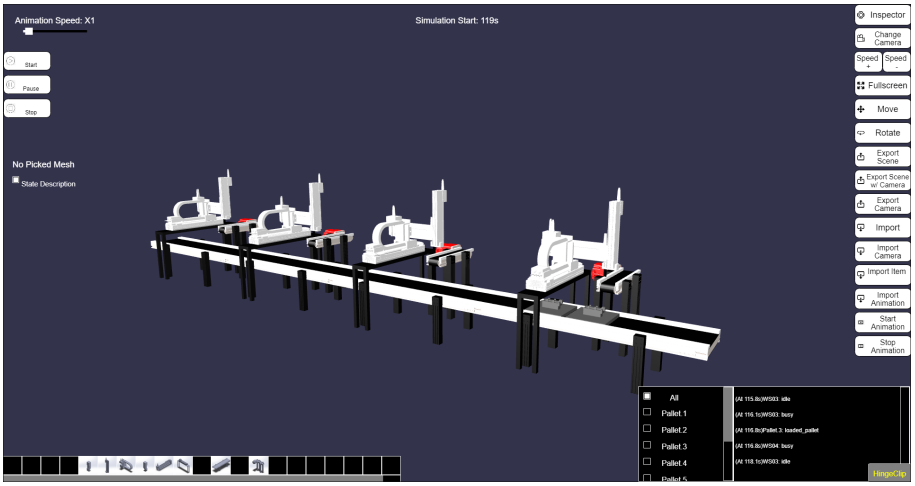


Figure 74 The Screenshot of the Moment when the Pallet is in the Buffer.

6.2 LIMITATIONS

After exploring new functionality some limitations have been met:

- Depending on simulation model settings (arrival and processing times) there can be overlapping in the visualization since capacity of buffers is not enough (requires at least 2 in some cases). This limitation can be overcome using another simulation tool which supports logging events in the buffer.
- Depending on the number of nodes in the JSON file, parsing time can be high which leads to inconvenient usage of the application since waiting time is too high. Decreasing number of

nodes using grouping algorithm for events of the same node can overcome this limitation. Also, scene can be split into two or more parts so this can reduce number of nodes in each JSON file.

- Depending on simulation model settings (arrival and processing times) and maximum number of meshes in the scene it cannot be possible to reuse the same mesh since timespan between timestamp of job id_i and timestamp of job id_{i+N} entering the scene is too short. Possible solution can be to increase and maximum number of meshes which can lead to increase of parsing time. Another possible solution is to apply another mapping rule.

CHAPTER 7. CONCLUSIONS

In this paper, the idea, by the project VLFT, of decoupling software tool into three parts has been enriched mainly in two stages which are 3D visualization stage and data processing stage. For the 3D visualization stage, based on the previous work in VE.bjs, the animation functionalities of entities include different events during the animation have been developed and also some functionalities in GUI have been added to make the application more user-friendly. For the data processing stage, the algorithm DP.py has been developed with the programming language Python which is mainly response for analysing and translating the data in CSV file from Java Modelling Tools in the simulation stage into the JSON format for importing into the VE.bjs in 3D visualization stage.

After having done the developments in data processing stage and 3D visualization stage, a model has been built in both JMT and VE.bjs with one buffer capacity for each working station. The validation has been done with the model to test the running smoothness and robustness. During doing the validation, the result came out positively but still there are some potential improvements can be done for the future work.

7.1 FUTURE WORK

In the simulation stage, it is possible to use other simulation software tools for presenting different industrial cases with different types of manufacturing systems such as job shop, flow lines, etc.

For the data processing stage, the Python script can be adopted for multiple simulators and be able to process with more kinds of input and output format. Also, another improvement can be developing an algorithm to automatically build the model in the simulation stage according to the model has been built in the 3D visualization stage and also in the opposite direction.

For the 3D visualization stage, the improvement could be to solve the problem for exporting the scene from the application since for now, the functionality for exporting the model which is built in the application hasn’t fully realized. Another optimization could be to optimize the data parsing structures to shorten the time for parsing data from the import files. Also, more functionalities can be done in several aspects, such as adding the camera on the entity which can move with the entity during animation, more descriptions of the state such as various failure modes and possible improvements in representing the states of entities with such as traffic lights or some other visualization methods.

REFERENCE

[1] M. C. F. Souza, M. Sacco, and A. J. V. Porto, 'Virtual manufacturing as a way for the factory of the future', *J. Intell. Manuf.*, vol. 17, no. 6, pp. 725–735, Dec. 2006, doi: 10.1007/s10845-006-0041-1.

[2] J. Zhou, P. Li, Y. Zhou, B. Wang, J. Zang, and L. Meng, 'Toward New-Generation Intelligent Manufacturing', *Engineering*, vol. 4, no. 1, pp. 11–20, Feb. 2018, doi: 10.1016/j.eng.2018.01.002.

[3] G. Zhou, C. Zhang, Z. Li, K. Ding, and C. Wang, 'Knowledge-driven digital twin manufacturing cell towards intelligent manufacturing', *Int. J. Prod. Res.*, vol. 58, no. 4, pp. 1034–1051, Feb. 2020, doi: 10.1080/00207543.2019.1607978.

[4] F. Tao, Q. Qi, A. Liu, and A. Kusiak, 'Data-driven smart manufacturing', *J. Manuf. Syst.*, vol. 48, pp. 157–169, Jul. 2018, doi: 10.1016/j.jmsy.2018.01.006.

[5] Q. Qi, D. Zhao, T. W. Liao, and F. Tao, 'Modeling of Cyber-Physical Systems and Digital Twin Based on Edge Computing, Fog Computing and Cloud Computing Towards Smart Manufacturing', presented at the ASME 2018 13th International Manufacturing Science and Engineering Conference, Sep. 2018, doi: 10.1115/MSEC2018-6435.

[6] X. Chen, S. Zhang, and J. M. P. Geraedts, 'Guest Editorial Focused Section on Sensing and Perception Systems for Intelligent Manufacturing (SPIM)', *IEEEASME Trans. Mechatron.*, vol. 23, no. 3, pp. 983–985, Jun. 2018, doi: 10.1109/TMECH.2018.2837008.

[7] C. Zhang, G. Zhou, Q. Lu, and F. Chang, 'Graph-based knowledge reuse for supporting knowledge-driven decision-making in new product development', *Int. J. Prod. Res.*, vol. 55, no. 23, pp. 7187–7203, Dec. 2017, doi: 10.1080/00207543.2017.1351643.

[8] B. Chen, J. Wan, L. Shu, P. Li, M. Mukherjee, and B. Yin, 'Smart Factory of Industry 4.0: Key Technologies, Application Case, and Challenges', *IEEE Access*, vol. 6, pp. 6505–6519, 2018, doi: 10.1109/ACCESS.2017.2783682.

[9] '501321main_TA11-MSITP-DRAFT-Nov2010-A1.pdf'. Accessed: Feb. 16, 2021. [Online]. Available: https://www.nasa.gov/pdf/501321main_TA11-MSITP-DRAFT-Nov2010-A1.pdf.

[10] W. Terkaj, T. Tolio, and M. Urgo, 'A virtual factory approach for in situ simulation to support production and maintenance planning', *CIRP Ann.*, vol. 64, no. 1, pp. 451–454, Jan. 2015, doi: 10.1016/j.cirp.2015.04.121.

[11] M. Sacco, P. Pedrazzoli, and W. Terkaj, 'VFF: Virtual Factory Framework', in *2010 IEEE International Technology Management Conference (ICE)*, Jun. 2010, pp. 1–8, doi: 10.1109/ICE.2010.7477041.

[12] E. Yildiz, C. Møller, and A. Bilberg, 'Virtual Factory: Digital Twin Based Integrated Factory Simulations', *Procedia CIRP*, vol. 93, pp. 216–221, Jan. 2020, doi: 10.1016/j.procir.2020.04.043.

[13] W. Terkaj and M. Urgo, 'Virtual Factory Data Model to support Performance Evaluation of Production Systems', Jul. 2012, vol. 886, pp. 44–58.

[14] K. Iwata, M. Onosato, K. Teramoto, and S. Osaki, 'A Modelling and Simulation Architecture for Virtual Manufacturing Systems', *CIRP Ann.*, vol. 44, no. 1, pp. 399–402, Jan. 1995, doi: 10.1016/S0007-8506(07)62350-6.

[15] B. Kádár, A. Lengyel, L. Monostori, Y. Suginishi, A. Pfeiffer, and Y. Nonaka, 'Enhanced control of complex production structures by tight coupling of the digital and the physical worlds', *CIRP Ann.*, vol. 59, no. 1, pp. 437–440, Jan. 2010, doi: 10.1016/j.cirp.2010.03.123.

[16] ‘Industry Foundation Classes’, *Wikipedia*. Jan. 23, 2021, Accessed: Mar. 22, 2021. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Industry_Foundation_Classes&oldid=1002190886.
[17] C. I. Schlenoff, M. Gruninger, F. Tissot, J. Valois, J. Lubell, and J. W. Lee, ‘The Process Specification Language (PSL) Overview and Version 1.0 Specification’, Feb. 2000, Accessed: Mar. 22, 2021. [Online]. Available: https://www.nist.gov/publications/process-specification-language-psl-overview-and-version-10-specification.
[18] ‘Process Specification Language’, *Wikipedia*. Sep. 24, 2018, Accessed: Mar. 22, 2021. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Process_Specification_Language&oldid=860941039.
[19] F. Zhou *et al.*, ‘A survey of visualization for smart manufacturing’, *J. Vis.*, vol. 22, no. 2, pp. 419–435, Apr. 2019, doi: 10.1007/s12650-018-0530-2.
[20] U. H. Govindarajan, A. J. C. Trappey, and C. V. Trappey, ‘Immersive Technology for Human-Centric Cyberphysical Systems in Complex Manufacturing Processes: A Comprehensive Overview of the Global Patent Profile Using Collective Intelligence’, *Complexity*, vol. 2018, p. e4283634, Feb. 2018, doi: 10.1155/2018/4283634.
[21] W. Mueller-Wittig, R. Jegathese, Meehae Song, J. Quick, Haibin Wang, and Yongmin Zhong, ‘Virtual factory - highly interactive visualisation for manufacturing’, in *Proceedings of the Winter Simulation Conference*, Dec. 2002, vol. 2, pp. 1061–1064 vol.2, doi: 10.1109/WSC.2002.1166356.
[22] B. Kádár, W. Terkaj, and M. Sacco, ‘Semantic Virtual Factory supporting interoperable modelling and evaluation of production systems’, *CIRP Ann.*, vol. 62, no. 1, pp. 443–446, Jan. 2013, doi: 10.1016/j.cirp.2013.03.045.
[23] S. Jain, G. Shao, and S.-J. Shin, ‘Manufacturing data analytics using a virtual factory representation’, *Int. J. Prod. Res.*, vol. 55, pp. 1–15, Jul. 2017, doi: 10.1080/00207543.2017.1321799.
[24] ‘Plant Simulation and Throughput Optimization’, *Siemens Digital Industries Software*. https://www.plm.automation.siemens.com/global/en/products/manufacturing-planning/plant-simulation-throughput-optimization.html (accessed Mar. 23, 2021).
[25] ‘Synchronizing physical and digital factory: benefits and technical challenges’, *Procedia CIRP*, vol. 79, pp. 472–477, Jan. 2019, doi: 10.1016/j.procir.2019.02.125.
[26] B. Schleich, N. Anwer, L. Mathieu, and S. Wartzack, ‘Shaping the digital twin for design and production engineering’, *CIRP Ann.*, vol. 66, no. 1, pp. 141–144, 2017, doi: 10.1016/j.cirp.2017.04.040.
[27] V. Kuts, G. Modoni, W. Terkaj, T. Tähemaa, M. Sacco, and T. Otto, ‘Exploiting Factory Telemetry to Support Virtual Reality Simulation in Robotics Cell’, Jun. 2017, doi: 10.1007/978-3-319-60922-5_16.
[28] F. Tao, J. Cheng, Q. Qi, M. Zhang, H. Zhang, and F. Sui, ‘Digital twin-driven product design, manufacturing and service with big data’, 2018, doi: 10.1007/S00170-017-0233-1.
[29] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, ‘Edge Computing: Vision and Challenges’, *IEEE Internet Things J.*, 2016, doi: 10.1109/JIOT.2016.2579198.
[30] ‘A+Brief+Comparison+of+Three+Simulation+Sofwares+-+Arena,+Anylogic,+Flexsim+by+Dr.+Alvin+Ang.pdf’. Accessed: Mar. 16, 2021. [Online]. Available:

https://static1.squarespace.com/static/5530dddfe4b0679504639dc1/t/5ef080d8748a14243021b27c/1592819944067/A+Brief+Comparison+of+Three+Simulation+Sofwares+-+Arena%2C+Anylogic%2C+Flexsim+by+Dr.+Alvin+Ang.pdf.

[31] '10 Best Simulation Software [CY] (Free and Paid)', *Woofresh*, Nov. 25, 2016. https://woofresh.com/simulation-software/ (accessed Mar. 16, 2021).

[32] 'Features'. https://www.anylogic.com/features/ (accessed Mar. 16, 2021).

[33] D. Unrau, 'Survey and Comparison of Modelling Software', p. 52.

[34] 'Online Process Simulator (OPS) - Descreye Solutions'. https://www.descreye.com/blog/a-comparison-of-discrete-event-simulation-software-2019/ (accessed Mar. 16, 2021).

[35] V. Havard, B. Jeanne, M. Lacomblez, and D. Baudry, 'Digital twin and virtual reality: a co-simulation environment for design and assessment of industrial workstations', *Prod. Manuf. Res.*, vol. 7, no. 1, pp. 472–489, Jan. 2019, doi: 10.1080/21693277.2019.1660283.

[36] L. Gong, J. Berglund, Å. Fast-Berglund, B. Johansson, Z. Wang, and T. Börjesson, 'Development of virtual reality support to factory layout planning', *Int. J. Interact. Des. Manuf. IJIDeM*, vol. 13, no. 3, pp. 935–945, Sep. 2019, doi: 10.1007/s12008-019-00538-x.

[37] 'The 3 Types of Virtual Reality', *Heizenrader*, Sep. 11, 2019. https://heizenrader.com/the-3-types-of-virtual-reality/ (accessed Mar. 17, 2021).

[38] C. Finch, 'Manufacturing with VR Becoming a (Virtual) Reality', *QAD Blog*, Sep. 25, 2018. https://www.qad.com/blog/2018/09/manufacturing-vr-becoming-virtual-reality (accessed Mar. 17, 2021).

[39] turvoinc, 'Virtual Reality in Manufacturing: 5 Benefits', *Transportation Management Company | Cerasis*, Mar. 14, 2016. https://cerasis.com/virtual-reality-in-manufacturing/ (accessed Feb. 08, 2021).

[40] 'Benefits Of VR Technology In The Manufacturing Industry', *4Experience Virtual Reality Studio*, Jul. 07, 2020. https://4experience.co/benefits-of-vr-technology-in-the-manufacturing-industry/ (accessed Mar. 17, 2021).

[41] C. Siedler, M. Glatt, P. Weber, A. Ebert, and J. C. Aurich, 'Engineering changes in manufacturing systems supported by AR/VR collaboration', *Procedia CIRP*, vol. 96, pp. 307–312, Jan. 2021, doi: 10.1016/j.procir.2021.01.092.

[42] Y. Shi, J. Du, S. Lavy, and D. Zhao, 'A Multiuser Shared Virtual Environment for Facility Management', *Procedia Eng.*, vol. 145, pp. 120–127, 2016, doi: 10.1016/j.proeng.2016.04.029.

[43] A. C. A. Mól, C. A. F. Jorge, and P. M. Couto, 'Using a Game Engine for VR Simulations in Evacuation Planning', *IEEE Comput. Graph. Appl.*, vol. 28, no. 3, pp. 6–12, May 2008, doi: 10.1109/MCG.2008.61.

[44] 'Unity vs Unreal comparison: Choose the best engine for you - 4Experience', *4Experience Virtual Reality Studio*, Jan. 25, 2021. https://4experience.co/unity-vs-unreal-comparison-choose-the-best-engine-for-you/ (accessed Mar. 17, 2021).

[45] Mahesa） 拉卡（Raka, 'Unity and Unreal comparison', *Packt Hub*, Jan. 26, 2018. https://hub.packtpub.com/unity-and-unreal-comparison/ (accessed Mar. 17, 2021).

[46] 'Unity3D vs Unreal Engine - Which is the right engine for you? | Sliced Bread Animation', Sep. 28, 2020. https://sbanimation.com/unity3d-vs-unreal-engine-which-is-the-right-engine-for-you/ (accessed Mar. 17, 2021).

[47] 'WebGL', *Wikipedia*. Jan. 31, 2021, Accessed: Feb. 16, 2021. [Online]. Available: https://it.wikipedia.org/w/index.php?title=WebGL&oldid=118357627.

[48] S. Bahor and B. Ramic-Brkic, ‘HTML5/WebGL vs Flash in 3D Visualisation’, Jan. 2013.
[49] A. Álvarez, C. Zapata, and E. Ramírez, ‘EZ3.js: A robust and flexible WebGL-based 3D engine’, in *2016 XLII Latin American Computing Conference (CLEI)*, Oct. 2016, pp. 1–9, doi: 10.1109/CLEI.2016.7833404.
[50] ‘20 Open Source WebGL Frameworks’, *MeDEVEL.com*, Jun. 25, 2019. https://medevel.com/16-webgl-opensource-frameworks/ (accessed Feb. 05, 2021).
[51] ‘Top 6 JavaScript and HTML5 game engines’, *LogRocket Blog*, Sep. 15, 2020. https://blog.logrocket.com/top-6-javascript-and-html5-game-engines/ (accessed Feb. 17, 2021).
[52] ‘VLFT | Virtual Learning Factory Toolkit’. https://www.vlft.eu/index.html (accessed Mar. 27, 2021).
[53] ‘Java Modelling Tools - Introduction’. http://jmt.sourceforge.net/ (accessed Feb. 13, 2021).
[54] G. Serazzri, G. Casale, M. Bertoli, G. Serazzri, G. Casale, and M. Bertoli, ‘Java Modelling Tools: an Open Source Suite for Queueing Network Modelling andWorkload Analysis’, in *Third International Conference on the Quantitative Evaluation of Systems - (QEST’06)*, Sep. 2006, pp. 119–120, doi: 10.1109/QEST.2006.22.
[55] G. Casale, G. Serazzi, and M. Bertoli, ‘for JMT version 1.1.0 and above’, p. 231.
[56] ‘Babylon.js Playground’. https://playground.babylonjs.com/ (accessed Feb. 06, 2021).