

Teemu Saaranen

# UTILIZATION OF SIMULATIONS FOR THE DEVELOPMENT OF OPERATOR ASSISTANT SYSTEMS IN FOREST MACHINES

Master of Science Thesis  
Faculty of Engineering and Natural Sciences  
Examiner: Prof. Jouni Mattila  
September 2023

# ABSTRACT

Teemu Saaranen: Utilization of simulations for the development of operator assistant systems in forest machines  
Master of Science Thesis  
Tampere University  
Master's Degree Programme in Automation Engineering  
September 2023

---

Utilization of simulations for the development of operator assistant systems in forest machines are studied in this Master's Thesis. The aim of this thesis is to create a basis for Ponsse's operator assistant systems simulation by surveying and testing the suitability of different simulation software for Ponsse's sensor simulation needs.

Several simulation tools and software options were surveyed for the simulations of operator assistant systems and the sensor simulation process provided by Mevea was chosen for a more detailed analysis. In this thesis, the creation process of the sensor simulation combination consisting of Mevea, Unity and ROS2 is clarified in detail and the suitability of this simulation process to Ponsse's requirements is evaluated using relevant use cases.

In addition to Mevea's sensor simulation process, the thesis also evaluates the suitability of alternative sensor simulation combinations for Ponsse's needs. Alternative sensor simulation tools and software are Creanex's sensor simulation combination in a training simulator and simulation software from Siemens, Cognata and CARLA for the development of advanced driver assistance systems and autonomous driving.

Based on the surveying of different software and simulation processes, the sensor simulation combination that meets Ponsse's requirements should include a simulation process and a visual process. Game engines that provide immersive renderings require a suitable physics engine alongside them that the simulation combination can be utilized in practical R&D work and it brings enough benefits for Ponsse. The results of the thesis show that the game engine Unity offers a versatile and efficient tool for creating forest environments, developing autonomous systems and the possibility to integrate the physics of the forest machine into the simulation combination using built-in or external physics engine. Physics engine studies show that Mevea brings benefits for Ponsse with its advanced real-time physics engine which enables the development and testing of new features of forest machines. On the other hand, the physics engines and sensor simulation platforms focused on the automotive industry do not bring enough benefits for Ponsse's forest industry needs.

Keywords: simulation, forest machine, operator assistant systems, LiDAR

The originality of this thesis has been checked using the Turnitin OriginalityCheck service.

# TIIVISTELMÄ

Teemu Saaranen: Simulointien hyödyntäminen metsäkoneen kuljettajaa avustavien järjestelmien kehittämisessä  
Diplomityö  
Tampereen yliopisto  
Automaatiotekniikan DI-ohjelma  
Syyskuu 2023

---

Tässä diplomityössä tutkitaan simulointien hyödyntämismahdollisuuksia metsäkoneen kuljettajaa avustavien järjestelmien kehittämisessä. Työn tavoitteena on luoda pohja Ponssen kuljettajaa avustavien järjestelmien simuloinnille kartoittamalla ja testaamalla eri simulointiohjelmistojen soveltuvuutta Ponssen sensorisimulointitarpeisiin.

Kuljettajaa avustavien järjestelmien simulointeja varten kartoitettiin useita eri simulointityökaluja sekä ohjelmistovaihtoehtoja, joiden joukosta tarkempaan analyysiin valikoitui Mevean tarjoama sensorisimulointiprosessi. Työssä käydään läpi Meveasta, Unitysta sekä ROS2:sta koostuvan sensorisimulointikokonaisuuden muodostaminen, ja tämän simulointiprosessin soveltuvuutta Ponssen tarpeisiin arvioidaan LiDAR-käyttötapausten avulla.

Mevean sensorisimulointiprosessin lisäksi työssä arvioidaan myös vaihtoehtoisten sensorisimulointikokonaisuuksien soveltuvuutta Ponssen tarpeisiin. Vaihtoehtoisista sensorisimulointityökaluista ja ohjelmistoista käsitellään Creanexin koulutussimulaattorissa toteuttama sensorisimulointikokonaisuus sekä Siemensin, Cognatan ja CARLA:n kuljettajaa avustavien järjestelmien ja autonomisen ajamisen kehittämiseen tarkoitetut simulointiohjelmistot.

Eri ohjelmistojen ja simulointiprosessien kartoituksen perusteella voidaan sanoa, että Ponssen vaatimukset täyttävän sensorisimulointikokonaisuuden tulee sisältää simulointi- ja visuaaliprosessi. Vaikuttavia renderöintejä tarjoavat pelimoottorit vaativat rinnalleen sopivan fysiikkamoottorin, jotta simulointikokonaisuutta voidaan hyödyntää käytännön tuotekehitystyössä ja se tuottaa riittävästi etuja Ponsselle. Työn tulokset osoittavat, että pelimoottoreista Unity tarjoaa monipuolisen ja tehokkaan työkalun metsäympäristöjen luomiseen, autonomisten ominaisuuksien kehittämiseen sekä mahdollisuuden integroida simulointikokonaisuuteen metsäkoneen fysiikkaa sisäinrakennetun tai erillisen fysiikkamoottorin avulla. Tutkituista fysiikkamoottoreista Mevea tarjoaa Ponsselle etuja edistyneellä reaaliaikaisella fysiikkamoottorilla, joka mahdollistaa metsäkoneiden uusien ominaisuuksien kehittämisen ja testaamisen. Sen sijaan autoteollisuuteen panostavat fysiikkamoottorit sekä sensorisimulointialustat eivät tuota riittävästi etuja Ponssen metsäteollisuuden tarpeisiin.

Avainsanat: simulointi, metsäkone, kuljettajaa avustavat järjestelmät, LiDAR

Tämän julkaisun alkuperäisyys on tarkastettu Turnitin OriginalityCheck –ohjelmalla.

# **PREFACE**

This Master's Thesis is done in cooperation with Ponsse. I would like to thank Ponsse for providing this opportunity to work with such an interesting topic. I am truly grateful to my personal supervisor Taneli Heikkilä for his advice and support. I also want to thank my supervisor Jouni Mattila from Tampere University for all the support and feedback during this thesis. Last but not least, I want to thank my family and friends for supporting me during my studies.

Tampere, 30.9.2023

Teemu Saaranen

# CONTENTS

1. INTRODUCTION .....	1
2. THEORETICAL BACKGROUND.....	3
2.1 Operator assistant systems in forest machines .....	3
2.2 Model-based design.....	4
2.3 Simulation .....	5
2.4 Modeling and simulation as a part of design process .....	6
2.5 MIL, SIL and HIL simulation .....	7
3. SIMULATION PROCESS.....	9
3.1 Software selection.....	9
3.2 Simulation process with chosen software.....	10
3.2.1 Mevea.....	11
3.2.2 Unity .....	13
3.2.3 ROS2.....	15
3.2.4 RViz.....	17
3.2.5 Interfaces between simulation tools .....	18
4. LIDAR SIMULATION USE CASES WITH CHOSEN TOOLS .....	21
4.1 Tree detection.....	22
4.2 LiDAR types.....	23
4.3 LiDAR locations .....	24
4.4 The effect of weather conditions.....	25
5. ALTERNATIVE SENSOR SIMULATION TOOLS AND PROCESSES.....	27
5.1 Siemens Prescan.....	27
5.2 Creanex .....	30
5.3 Cognata .....	32
5.4 CARLA.....	33
6. DISCUSSION.....	36
6.1 Further development.....	42
7. CONCLUSIONS.....	45
REFERENCES.....	47

# LIST OF SYMBOLS AND ABBREVIATIONS

ADAS	Advanced Driver Assistant Systems
API	Application Programming Interface
COM	Component Object Model
CPU	Central Processing Unit
DLL	Dynamic-Link Library
GPU	Graphics Processing Unit
GUI	Graphical User Interface
HIL	Hardware-in-the-Loop
LiDAR	Light Detection and Ranging
MIL	Model-in-the-Loop
MBD	Model-Based Design
RADAR	Radio Detecting and Ranging
RCL	ROS Client Library
ROS	Robot Operating System
RViz	ROS Visualization
SIL	Software-in-the-Loop
STL	Stereolithography

# 1. INTRODUCTION

Mechanized harvesting requires professional skills from the forest machine operator because most of the harvesting steps are still done manually by forest machine operator. However, modern forest machines already contain automation technology and more and more manual harvesting steps will be automated in the future. The general aim of automation and operator assistant systems in the forest machines is to improve productivity of harvesting and reduce the impact of competence of the operators on productivity. In addition, to achieve cost and safety benefits in harvesting, even more functions must be independent of the human operator. The importance of simulation is significant in product development when developing these operator assistant systems.

The purpose of simulation is to imitate the operation of a real-world process or system with models. Simulations have become a significant part of product development because simulations save costs and speed up the design and development process safely compared to building a real prototype.

The aim of this thesis is to survey the utilization of simulations for the development of operator assistant systems in forest machines. The main target of the thesis is to provide a guideline how operator assistant systems simulations will be done at Ponsse. Before this guideline can be done, it is needed to select feasible software, methods and cooperation partners. To demonstrate how guideline is defined, it is needed to draw simulation process workflow of different potential alternatives. After the guideline is done, it is needed to conclude what is required to do before guideline can be implemented to practical R&D work or does it bring enough benefits. A key software requirement in the simulation of operator assistant systems in forest machines is the ease of creating harvesting scenarios in a forest environment. The possibility to model the physics of forest machines and create different kinds of forest environments is necessary. It is also a high priority that the sensor simulation process produces accurate point cloud data even under challenging conditions. Various integration possibilities are also important requirements of the chosen software. The goal is that the software could be integrated with the control system, i.e. Simulink. Integration possibilities with test automation is also relevant. Therefore, real-time simulation is crucial. For future development, machine learning and autonomous capabilities are also essential requirements. The requirements related to the use of the software are a reasonable priced license and customer support.

The thesis structure is as follows: first the theoretical background of operator assistant systems and the reasons why they are developed with simulations are presented. Then, the selection process of the software chosen for this thesis is presented. After that, a simulation combination is created with the chosen simulation tools and the suitability of the tools of this simulation process for Ponsse's needs is evaluated with use cases. Also, alternative simulation tools and simulation processes are presented. Finally, discussion and conclusions are made based on the results of the simulation with chosen simulation tools and studies of alternative simulation tools.

## 2. THEORETICAL BACKGROUND

The present chapter will introduce essential information about operator assistant systems in forest machines, model-based design and simulation, as they are the foundation of this thesis. Theoretical perspective creates the foundation for the practical part of the thesis by explaining the definition, meaning and advantages and disadvantages of modeling and simulation. The theory answers the following question: Why are operator assistant systems developed with simulations?

### 2.1 Operator assistant systems in forest machines

Operator assistant systems in forest machines are systems that assist the operator in decision-making and execution during the harvesting process. Modern forest machines already contain functions that assist the operators, such as boom tip control, fault diagnostics, satellite navigation, communication and productivity monitoring. Despite all the automation efforts, modern forest machines are still at the lowest possible level of driving automation and there are no fully automated harvesting steps other than the processing of a flawless log. Professional operators are still needed to execute all other harvesting phases. [1]

However, there will be a need for operator assistant systems in the future because there is a global shortage of professional forest machine operators. With the operator assistant systems, machine productivity between professional and beginner operators can be closer to each other. Thus, the operator assistant systems improve forest machines productivity. In addition, operator assistant systems can reduce the operator's workload and the possibility of human errors. In this way, operator assistant systems also improve the safety of the working environment and reduce the strain on the forest machine and the possibility of crashes.

Due to the operator assistant systems advantages, manufacturers of forest machines will invest in product development and generalization of operator assistant systems in the near future. However, the development of operator assistant systems is not a simple process. For example, the development of operator assistant systems and autonomous forest machines is more complex than the development of an autonomous on-road vehicle because the operating environment of forest machines is challenging. The biggest challenges in the operating environment are the lack of traffic rules or signs to follow, the variability of the environment and weather conditions such as fog, mud and rain.

The importance of modelling and simulations is emphasized in the development of operator assistant systems for challenging conditions. It is more beneficial to do careful modeling and simulation before building a real prototype.

## 2.2 Model-based design

Model-based design (MBD), also known as model-based development, is a mathematical and visual method of addressing problems associated with designing complex systems in industries [2]. Model-based design includes good conditions for simulating different situations and testing the applications that are developed. In model-based design, engineers design and simulate the model using a model-based design tools and present the ideas behind the solution as a working model of the system [3]. A model-based design approach does not require complex application, component or manufacturer specific coding to fulfil requirements because the needed functionality can be described with more general functional blocks. Model-based design tools often generate code automatically based on the visual model. The purpose of the created model is to act like a real system and describe the behavior and problems of the real system. Developing a detailed, but not too detailed or complex, model ensures sufficient information for analysis of the system. Therefore, the purpose of the model must be clear before modeling.

One of the main reasons to use models instead of real system is acceleration of development process as failures and bugs are found early in the development process. Visual modeling enables quick testing of ideas until an optimal solution is found. The model can be run for simulation or system testing purposes at any step of the development process. There is a possibility of major changes at late project phases. This continuous verification and validation enable successful project management and reduction of product development risks. Therefore, designed systems are high-quality and they reach the market faster and end up costing less. This approach reduces the number of physical prototypes because engineers can verify and optimize the design of the system before prototyping in hardware. [3] [4] [5]

There are also some challenges with model-based design. Model-based design needs resources because the needed tools are not particularly inexpensive and processes creation takes many working hours. Model-based design might also be complex. Complex systems such as autonomous vehicles may require many subsystems and components which are difficult to fit together in single model.

## 2.3 Simulation

Simulation is the imitation of the operation of a real-world process or system on a computer over time [6]. Thus, simulations are used to understand systems and their behavior without a real prototype. Model-based design is based on the creation of a functional virtual environment, that enables the simulation and testing of different scenarios in different phases of software development. For example, there are limited possibilities for testing the control system of a forest machine in a real machine. Testing is also typically slow and if errors are detected in the control system during machine testing, it can cause major delays in development schedules. Therefore, simulation is a necessary problem-solving method for the solution of many real-world problems.

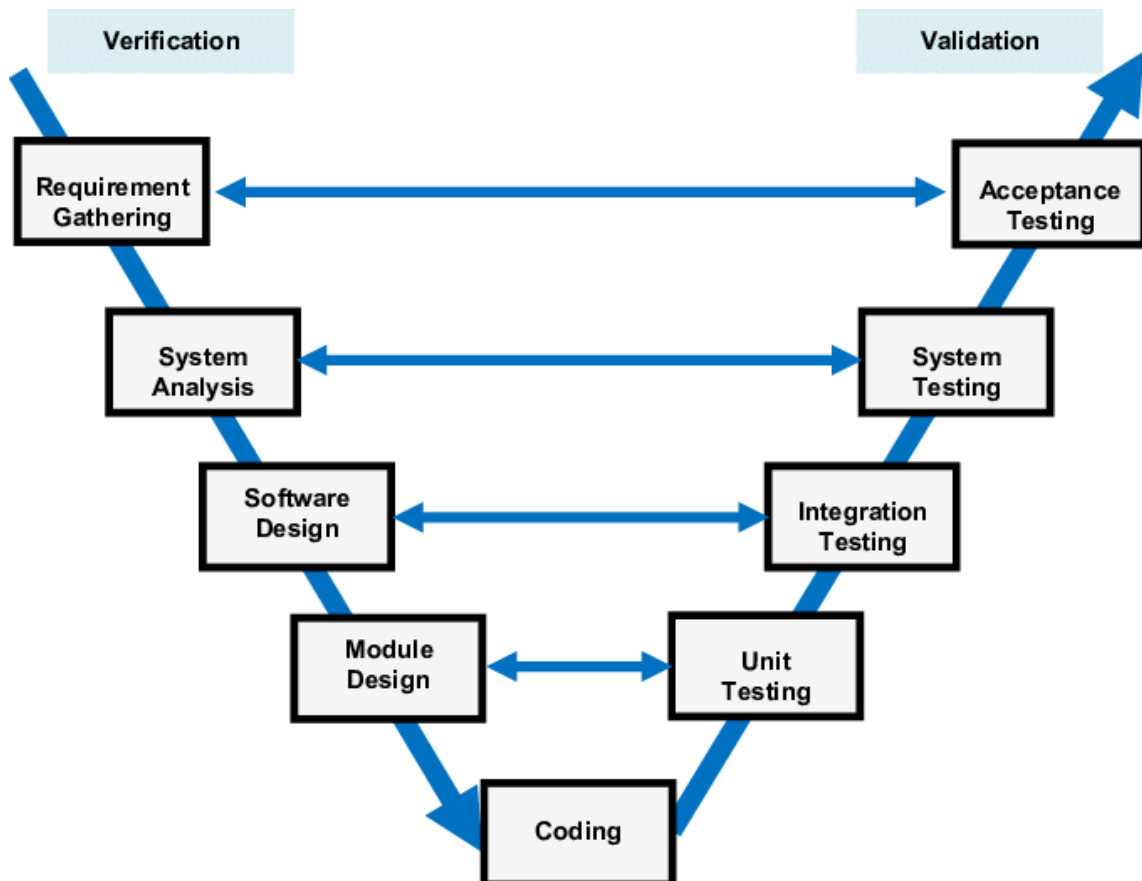
If the computer model runs at the same speed as the real physical system, the simulation is a real-time simulation. Real-time simulation produces the internal variables and output in same time as its physical counterpart. Real-time simulation maximizes the benefits of simulation and enables better innovativeness of the product.

One of the main advantages of simulation is safety. Forest machines are large and heavy and they operate in unstructured environments. There are numerous tests or training situations that involve human, material or environmental risks [7]. Therefore, a substantial part of testing or training must be done in a simulated environment. In a simulated environment, it is easier and safer to test how the system reacts to different or even unpredictable situations. It could be fatal to test these situations with physical prototypes. In addition, simulations reduce product development costs. Virtual production may be expensive but often cheaper than real production. Physical prototype test causes, for example, fuel and resource costs. And if the prototype test fails, further investment is required for each new test. Simulations and virtual tests provide a large amount of data available before building an expensive prototype. Due to these reasons, the current tendency is toward fewer tests and their replacement by simulation.

However, there are also some disadvantages in simulation. Simulation models always produce only an approximation of the studied system. Simulation results may not provide useful information about the actual system if the simulation model is not accurate enough. On the other hand, it can be a time-consuming and expensive process to make the simulation model accurate enough. [8]

## 2.4 Modeling and simulation as a part of design process

As presented in the previous chapters, modeling and simulation are part of the design process. The development life cycle of the system can be represented by a V-model. The structure of V-model is described in figure 1.



**Figure 1.** V-model of software development [9].

V-model is an approach to validating and verifying system development [9]. Simulation models are approximate imitations of real-world systems and they may not be accurate enough. Due to that, the models should be verified and validated. Verification is the evaluation process of the product development process to find out whether specified requirements meet. Validation is a process after the development process to find out whether the software meets the expectations and requirements of the customers and if it is suitable for the intended purpose. [10]

V-model contains the verification phases on the left side of the model and these are the design phases. Validation phases are on the right side of the model and these are the testing phases. Verification and validation process is joined by coding phase. The basic idea of the model is that each verification phase has a parallel validation counterpart and these phases will be done simultaneously. [9]

The verification phase begins with gathering the requirements for the system. The next phase is to analyze the requirements and figure out possibilities and techniques to implement requirements. For example, how the system components will interact with each other to run the entire system. Software design or high-level design phase is a technical phase where the architecture of the system is designed. For example, all the modules and their functionality, interface relationships, dependencies and technology details. After high-level design, the low-level design phase breaks the system into small modules. The operation of the system and each module is defined in detail. After the designing, the last verification phase is coding. Coding is done based on the requirements and the design of the components and modules. The coding phase can be done manually or with simulation software's automatic code generation. [9] [10]

After the coding has been done, V-model will process to do validation phases in reverse order. Unit testing is validation for each type of scenario that each module might face. Unit testing is code-level testing and is used to eliminate bugs at an early stage. After unit testing, the next phase is the integration test. This phase is used to test the communication and coexistence of internal modules within the system. The next phase is system testing and it is directly associated with the system design phase. System tests check the functionality and the communication of the entire system with external systems. The last phase of the V-model is acceptance testing. This phase is associated with the requirements analysis and involves testing the product in the user environment. This phase reveals the compatibility issues with the other systems available in the user environment. Also, non-functional issues such as load and performance defects are discovered in this phase. [9] [10]

All in all, V-model provides a systematic way to system development. V-model is the most suitable for small and medium-sized projects where requirements are clearly defined. Using the V-model is easy to execute because each phase has defined objectives and goals. Thus, each phase of the V-model is scheduled. There are also some disadvantages in the V-model. This approach is not flexible and is not good for complex projects. In addition, software development is done during the implementation phase, so early prototypes of the software are not produced. It is also challenging if there are any midway changes, then the test documents and the requirement documents must be updated. [10]

## **2.5 MIL, SIL and HIL simulation**

Simulation models must be tested between different testing phases to find both properly working features and undesirable features as the modelling process proceeds. There are

different testing methods for simulation models based on how they are used and what is needed from the simulation. This thesis focuses on Model-in-the-loop (MIL) and Software-in-the-loop (SIL) testing methods which are conducted in the early stages of the software development process. In the further development of the simulation models of this thesis, hardware-in-the-loop (HIL) testing method would become essential.

The first testing method is model-in-the-loop. The whole simulation model of the system is built and it is possible to simulate a complete environment to ensure that model meets the requirements. At this point, the functionality is verified without any physical components. [11]

Once the model has been verified in MIL simulation, the next testing method is software-in-the-loop. Everything in the SIL is implemented using software and specific hardware is not required to run simulation. SIL simulation represents the integration of compiled production source code into mathematical model simulation. SIL provides a practical virtual simulation environment for detailed development and testing of large and complex systems. [12] SIL testing is used in the unit and integration testing phases of the V-cycle and the simulation does not need to be achieved in real-time at these phases.

After MIL and SIL, hardware-in-the-loop simulation can be used as a last testing and verification method before testing with a machine in the real world. HIL is based on many of the same hardware components as the real machine. The software is running on the same computer hardware as in a real machine and the goal is to achieve real-time simulation. This testing method is the most expensive and complicated to build and maintain compared to MIL and SIL methods.

## 3. SIMULATION PROCESS

The present chapter introduces the selection process of suitable simulation software and the final combination of the chosen simulation tools for the detailed analysis. Also, the creation process of the chosen simulation combination is clarified in detail.

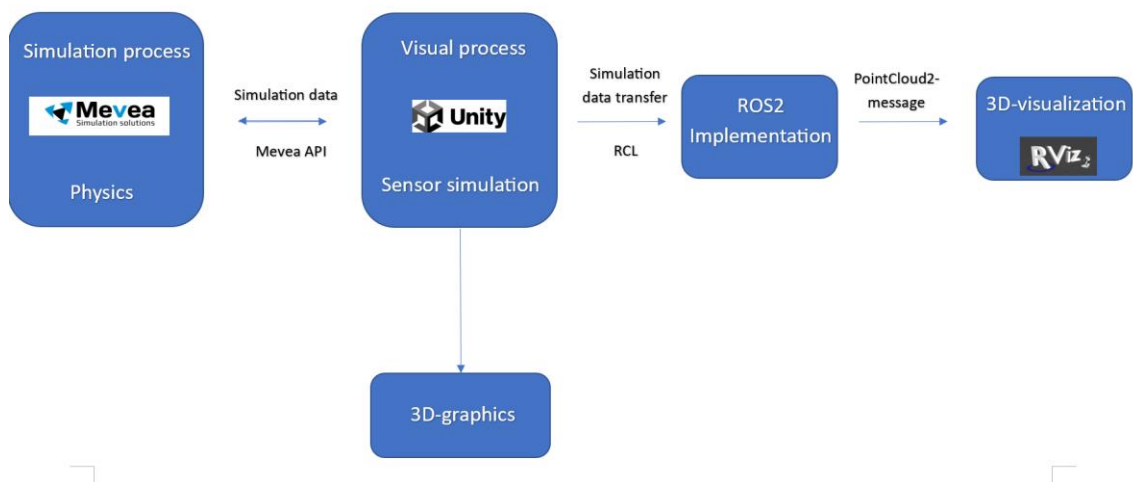
### 3.1 Software selection

The software selection process began by researching potential software suppliers with experience in sensor simulation and simulation of operator assistant systems for off-road machines. Most of the experienced simulation software companies focus their business mainly on the on-road vehicles but there are also some software companies on the market that focus on simulations of off-road machines. The possibilities and goals of the operator assistant system simulations were discussed with the potential partners. The most potential software suppliers that were surveyed were Ansys, Cognata, Creanex, Mevea and Siemens. All these software companies had a clear idea of operator assistant system simulations and they presented their demos and offered further support for operator assistant system simulations. In addition, a visit to the office of Creanex, Ponsse's long-term cooperation partner, provided an understanding of their vision and resources for simulations of operator assistant systems. During the software selection process, the strengths and the weaknesses of different software and the possibilities for simulating operator assistant systems became clearer.

Based on the survey, Mevea Simulation Software was chosen for the detailed analysis in this thesis. Mevea's experience in simulation of off-road machines and some collaborations with other forest machine manufacturers inspired confidence. Also, some mining industry companies, such as Sandvik and Normet, are using Mevea Simulation Software to develop their virtual prototypes. These mining companies and some of the port industry companies are also using Mevea's custom-built training simulators to train their operators. These companies have a positive experience with Mevea's simulation tools. [13] Ponsse was not very familiar with Mevea's simulation tools, except a few earlier test periods. Therefore, the potential of Mevea's simulation tools for simulating operator assistant systems was wanted to be tested and evaluated. Mevea presented their own sensor simulation demo which provided guidelines how Ponsse could create a simulation combination to simulate operator assistant systems. In addition, a reasonably priced trial license and ten-hour training course for Mevea Simulation Software users enticed to test suitability of the software for Ponsse's operator assistant system simulation needs.

### 3.2 Simulation process with chosen software

The purpose of the simulation process of operator assistant systems is to support Ponsse's product development and business operations. A functional simulation process can consist of several different simulation tools. The software of the simulation process and the interfaces between them must create a combination which produces benefits for Ponsse. One potential sensor simulation process created in cooperation with Mevea is explained in this chapter and the following subchapters. The process diagram of this simulation process is described in figure 2.



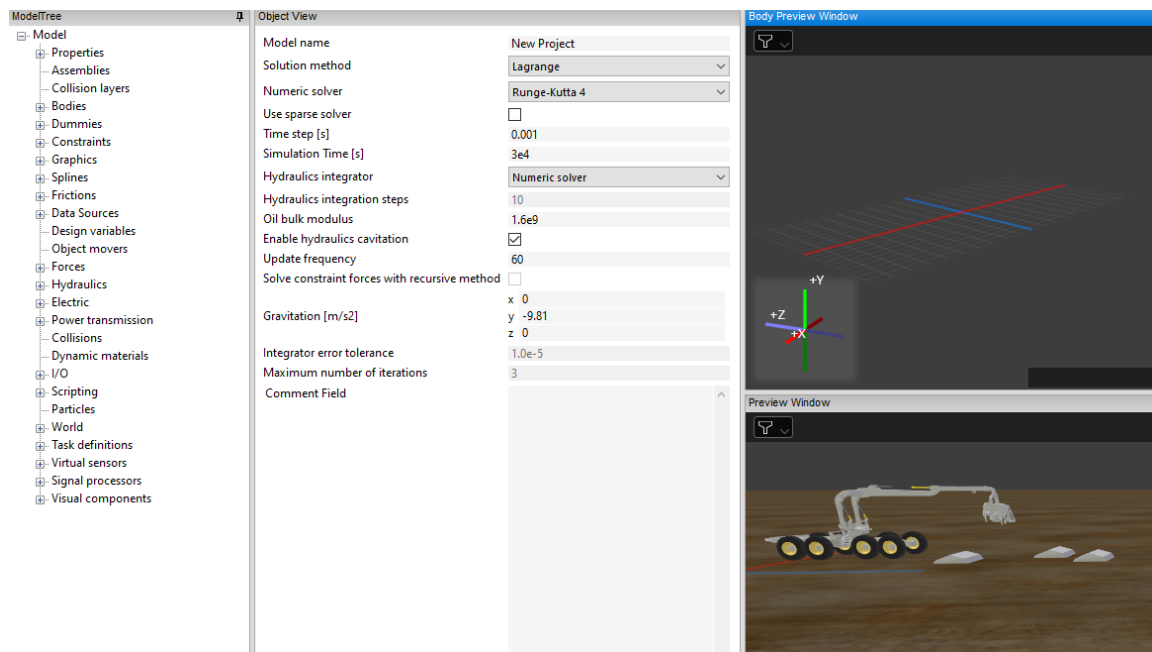
**Figure 2.** Sensor simulation process with Mevea.

The basic idea behind this sensor simulation process is that the process contains a separate simulation process and visualization process. The simulation process requires a physics engine and the visual process requires a game engine. In this case, Mevea's simulation tools and Mevea's own physics engine are used in the simulation process and Unity is a game engine in the visualization process. A physical model of the machine is created in the simulation process. The location and orientation data of the physical model can be exported to Unity's visual process using the Mevea API interface. The virtual sensors are included in the Unity project and the point cloud data is generated from the Unity visualization and the generated corresponding standard ROS2 message visualize it in RViz. The creation of this simulation process and the necessary simulation tools are described in more detail in the following subchapters. The suitability of this sensor simulation process for Ponsse's needs is evaluated in more detail using the use cases in chapter 4.

### 3.2.1 Mevea

Mevea Simulation Software is a real-time simulation tool for designing and testing new features of work machines. Mevea's own physics engine enables accurate simulation of mechanics, hydraulics, powertrain and the operating environment of the machine. The Mevea software tool is divided into three environments: Mevea Modeller, Mevea Solver and Mevea I/O toolbox. [14] In this thesis, only Mevea Modeller and Mevea Solver were used. Thus, Mevea I/O toolbox which connects external systems to simulation was not used.

A physical model according to the requirements is created in Mevea Modeller. Mevea Modeller enables designing mechanics, hydraulics and powertrain in the same platform. Mevea Modeller user interface is presented in figure 3.



**Figure 3.** Mevea Modeller user interface.

Mevea Modeller user interface consists of the Model Tree, Object View, Body Preview Window and Preview Window. Model Tree contains all model components which relations, values and other component parameters can be edited in Object View. The Body Preview Window shows the graphics of selected component and Preview window shows the model in a virtual world.

The modelling begins with the creation of graphics. Mevea Modeller is not a 3D modelling program, so all the graphics need to be imported, for example from Solidworks. The imported graphics file format must be light enough to run the simulation in real-time, for example STL file format. The imported body graphics include only the visual appearance.

Therefore, the mass properties such as mass, center of mass and moments and product of inertia must be set manually for bodies.

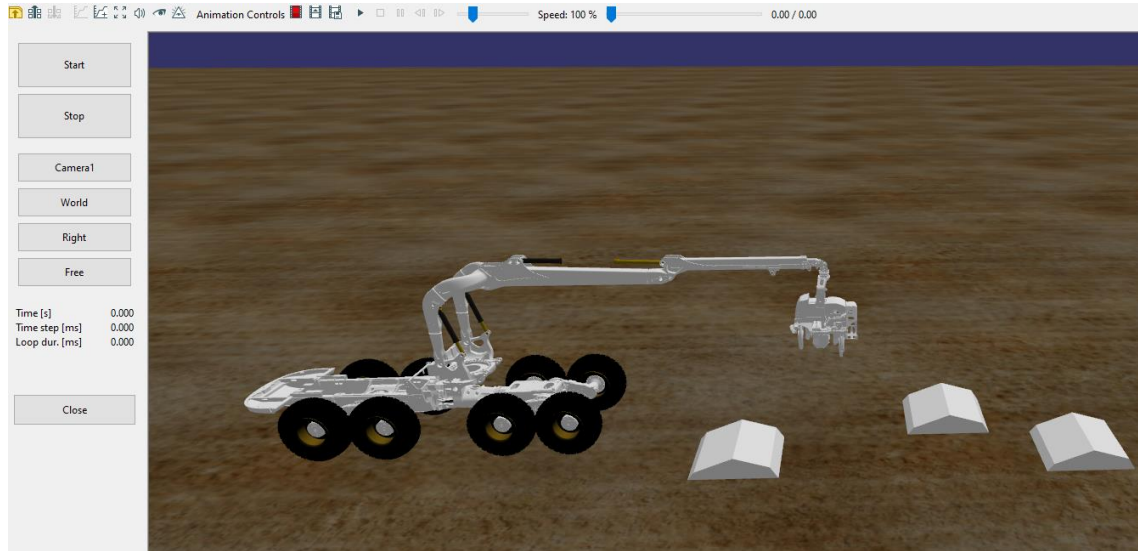
Visualization graphics help to place bodies and ensure the correct location of coordinate systems. The bodies are created in the order that they form a kinematic chain and the connection between the bodies and their locations in the system is clearly identified. The joints between bodies are called constraints. There is a selection of different joint types to define the number of degrees of freedom and which directions of rotational and translational movements are constrained.

After modelling the mechanics, the next step is to model simple hydraulics. Mevea Modeller supports hydraulic components that are enough to model mobile machines. Cylinders and piston rods are dummies. Dummies are separate parts connected to rigid bodies without joints. They are not connected to the kinematic chain but affect the mass and inertia properties of the body that they are attached to [15]. Dummies minimize the number of rigid bodies which makes the software's computation lighter and enables real-time simulation. Cylinder force must be defined as body-to-body force between two points in different bodies. Dummies are located between these points. Also, the maximum and minimum lengths of cylinder strokes must be defined to limit joints' movement. After the dummies, the parametrized volumes, hoses, pipes, and valves must be created. The last phase is to add control signals to control the machine. Inputs can be specified directly in Modeller and the inputs of the simulated model can be controlled in Solver with a keyboard control.

To create a drivable model, simple power transmission must also be created. Power transmission components to create in Modeller are clutch, gears, differentials, planet gears and brakes. The gear box type can be either automatic or manual and driving forward and backward is possible. The last step is to create tires and their contacts. Tires are dummies and their spring and damper constants parameters are important for achieving the proper flexibility behavior for the tire model in simulation [15]. Also, the chosen friction type affects the behavior of the tires. Finally, the operating environment of the machine, i.e. the contact graphic which has contact with the tires is chosen.

When a drivable physical model has been modelled, the model can be simulated to test if all the features meet the requirements. Models with several joints are recommended to simulate with Lagrange solution method. Recursive formulation provides more computational efficiency in real-time simulation which most cases requires but Lagrangian formulation is capable to handle all type of joints. [15]

Mevea Solver is used to simulate and visualize physical models created in Mevea Modeller. Mevea Solver enables running the simulation in real-time using Mevea simulation engine, even if the model is quite complex. Thus, can be verified if the desired results are obtained. Mevea Solver user interface is presented in figure 4.



*Figure 4. Mevea Solver user interface.*

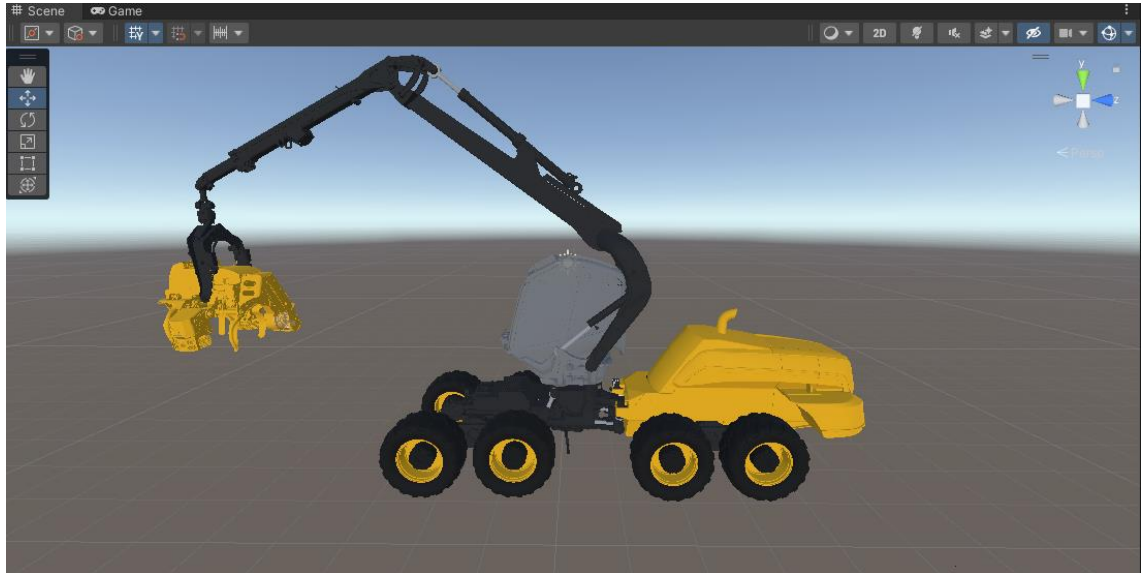
### 3.2.2 Unity

Unity is a real-time game engine developed by Unity Technologies used to create interactive 2D and 3D environments. In general, game engines are software used as a development tool for game development that provide ready-made basic functionalities, such as rendering, physics, sounds, scripting and animations [16]. Therefore, Unity is mainly used for game development, but nowadays Unity is also used for various simulations, such as this sensor simulation process.

Unity plays a significant role in this sensor simulation process. Firstly, Unity is used for graphical reasons. Although the physical model of the machine can be visualized in Mevea Solver, Unity creates a better, more immersive simulation from the graphical point of view. Unity is not only used for graphical reasons but also for virtual sensors. Virtual sensors are created and parametrized in Unity and they gather and generate data based on conditions surrounding them in Unity's environment.

Although Unity has its own physics system, for this type of sensor simulation combination it is recommended to use third-party software to imitate the machine's behavior. Therefore, all physics principles, operating environment and input controls of the simulation are derived from Mevea.

To combine Mevea's physical model and Unity's graphics, a graphical model of the machine must be created in Unity. All bodies and dummies used in Mevea must also be found in Unity. The graphical model of the machine created in Unity is presented in figure 5.



**Figure 5.** Graphical model of the machine in Unity.

When the physical and graphical model of the machine has been created, the next step is to combine them and ensure that the simulation data is transferred from the Mevea to the visualization of Unity in real-time. There is a Mevea API interface between software and this data transfer principle is explained in more detail in chapter 3.2.5. The combination of Mevea's physical model, forest environment and Unity's graphical model is presented in figure 6.



**Figure 6.** *Forest machine in Unity environment.*

Creation of the environment can be done either in Mevea Modeller or Unity. If the environment is created in Mevea, the advantage is the exact interaction between the machine and the environment. This enables, for example, realistic deformable soil behavior and contact with obstacles. Thus, in theory trees could be sawed and moved from one place to another. If the environment is created in Unity and not in the same software with the physical model, the interaction between the machine and the environment can be distorted. There are many tools for creating environments in Unity. If the goal is, for example, to detect only graphical obstacles with sensors without interaction between the machine and the obstacles, Unity's environment assets are sufficient in this process.

In this sensor simulation process, the sensors are the core of the simulation. These sensors are configured and attached to the desired body in Unity. The basic idea behind the sensors is that there is a ready-made depth camera rendering component in Unity that is also the basis for virtual LiDAR. The sensors defined in Unity are not connected to Mevea Solver. Sensor data is gathered in Unity and published in ROS2. The interface between Unity sensors and ROS2 is explained in more detail in chapter 3.2.5.

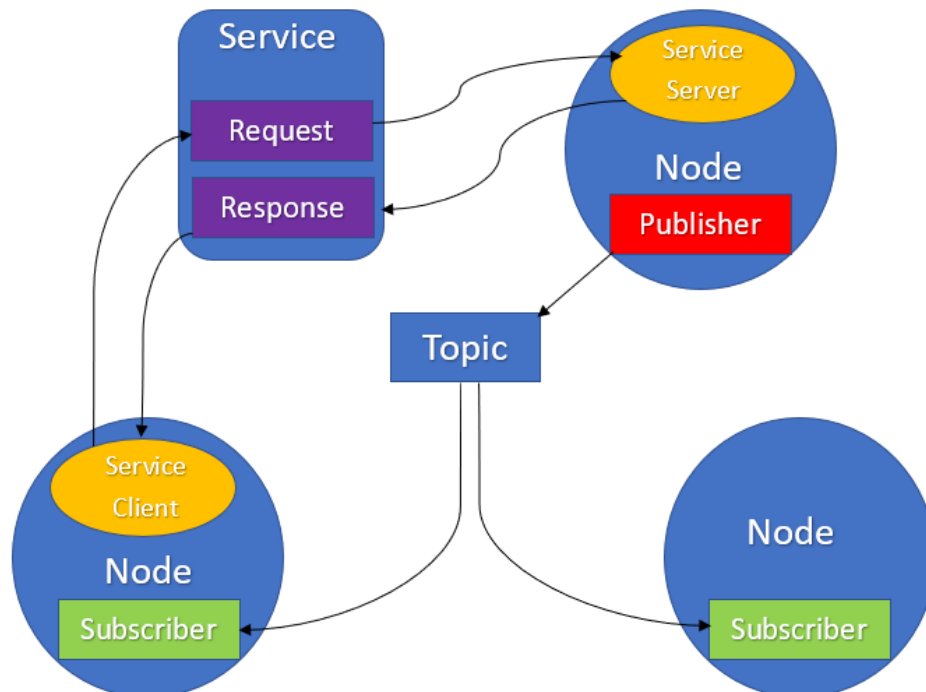
### **3.2.3 ROS2**

Robot Operating System 2 (ROS2) is an open-source middleware focused on the research and development of robotic applications. Despite its name, ROS2 does not provide a ready-made operating system for robots. ROS2 provides an interface for communication between processes on which different software components can be run. Real-

time performance support and the ability to run on various operating systems make ROS2 more advanced than its predecessor ROS. [17]

The operating principle of ROS2 is based on nodes that control the different functions of the system. These functions can be, for example, processing sensor data or moving the system to a certain destination. Communication between nodes is based on peer-to-peer communication with a publisher-subscriber model. Topics are used to send messages between nodes. Each topic has a name and message type that define the structure of the data. Topic can be meant for example for continuous sensor data. Nodes can publish their messages in the topics or subscribe to topics to receive messages. Nodes can have multiple publishers or subscribers that can be connected to multiple nodes. [18] [19]

ROS2 also contains other communication functions built around nodes and publisher-subscriber model. These functions are divided into services and actions. As the name suggests, the purpose of the services is to implement service. Instead of publishing and subscribing, one node is a server and the others are clients. The client node requests a service from the server node and the server node returns a response. Services are meant for example for running some calculations. [20] An example of nodes communication via topic and service is presented in figure 7.



**Figure 7.** ROS2 nodes communication with a topic and a service. Based on [17].

Actions are built using topics and services. Actions functionality is similar to services but for long running tasks. An action client node sends a goal to an action server node using a goal service. Then the action server acknowledges the goal, executes the action towards the goal and publishes its progression in the feedback topic. When the goal is achieved, the action server returns the result to the client server using the result service. [21]

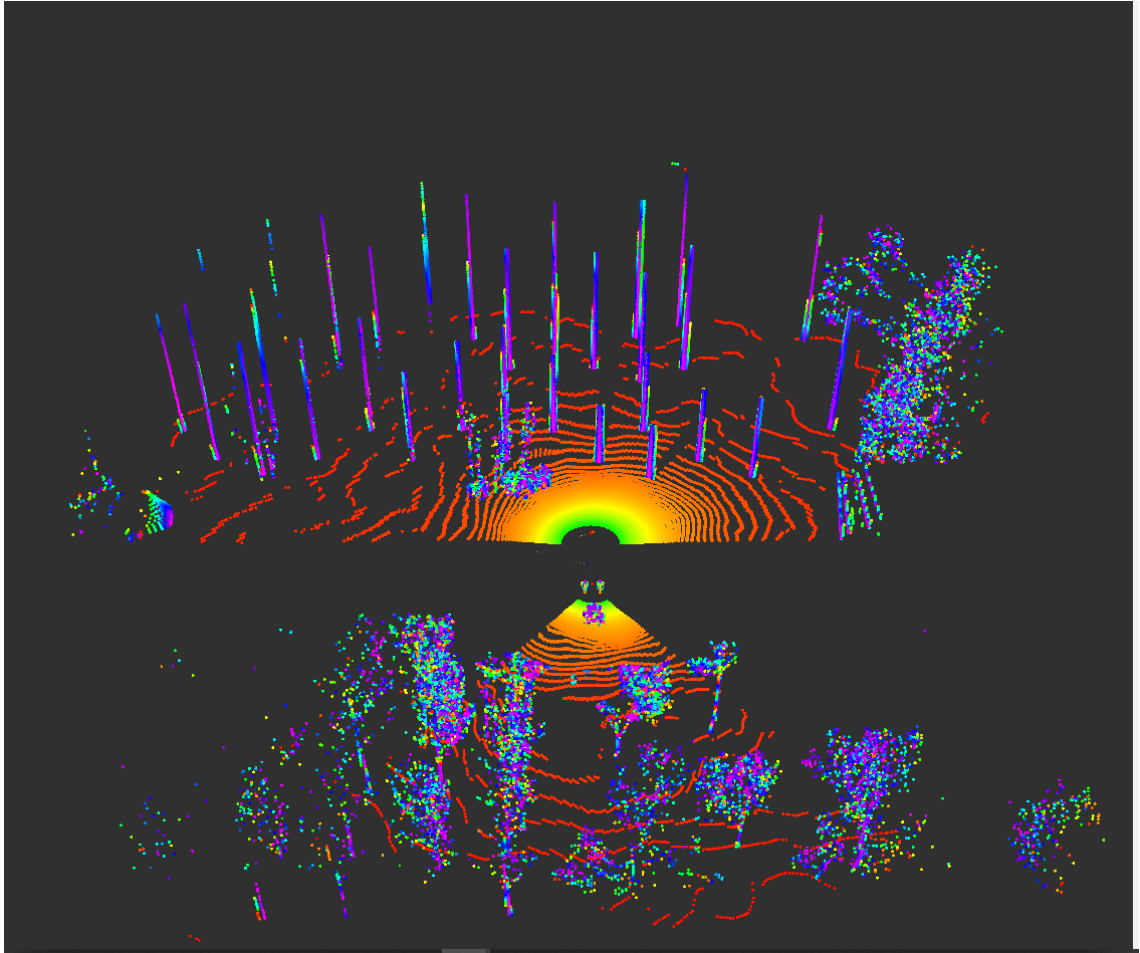
In this sensor simulation process, the purpose of ROS2 is to process the data gathered from Mevea Solver and Unity and publish it in ROS2 message format. ROS2 constantly receives coordinate data and interprets the location of the machine and the sensors attached to it. Thus, a real-time LiDAR point cloud can be obtained even if the machine is moving.

ROS2 messages are based on ROS Client Libraries (RCL). RCL is a collection of code for ROS programming and RCL is used to write ROS nodes and publish and subscribe to topics [22]. There are several nodes in ROS2 environment. Some of them are used for processing Mevea Solver data and others for processing Unity data. These Solver nodes and Unity nodes publish their ROS2 messages without communicating with each other. One of the messages published by Unity node is the pointcloud2 message which is based on the virtual LiDAR's point cloud data. Publishing this message enables visualization of LiDAR point cloud.

### **3.2.4 RViz**

ROS visualization (RViz) is a 3D visualization tool provided by ROS2. The original purpose of RViz is to visualize the robot with its environment and the robot's sensor data based on ROS messages. [23] The operating principle of RViz is to subscribe to relevant topics and draw a visualization of the published data.

This sensor simulation process utilizes the point cloud visualized by RViz based on relevant topics. Point cloud is a set of individual data points in 3D space. The points describe geospatial data where each point corresponds to an exact three-dimensional position relative to other points in the point cloud [24]. The point cloud of virtual LiDARs is visualized in figure 8.



**Figure 8.** Point cloud view of virtual LiDARs in RViz.

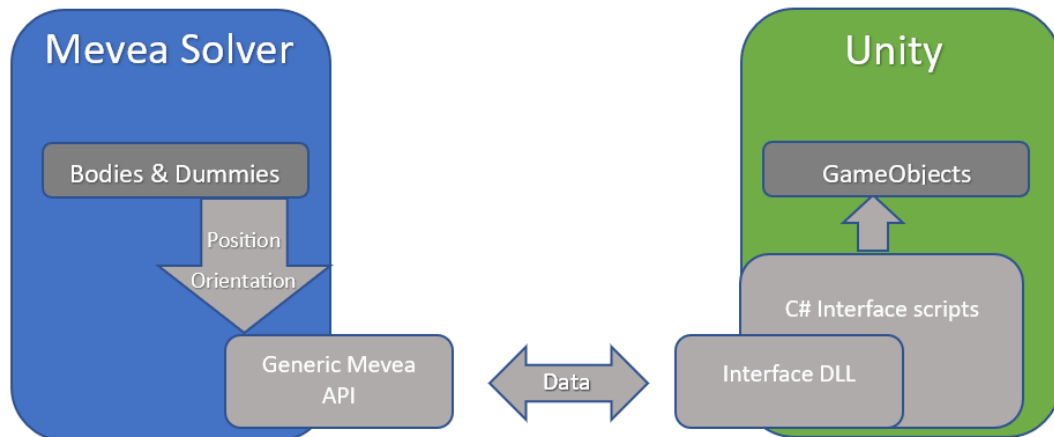
In the point cloud visualization, the points represent obstacles perceived by the virtual LiDAR sensors. Point cloud visualization is based on the message published by Unity node that is based on the pointcloud2 message generated from the point cloud data. Although this sensor simulation process focuses only on the visualization of the point cloud, the point cloud data can also be used for further development, for example the development of algorithms.

### **3.2.5 Interfaces between simulation tools**

All the software presented in chapters 3.2.1 – 3.2.4 are combined to communicate with each other using interfaces. Application programming interface (API) is like a bridge for data transfer between software. API enables an easier and automated way to update, manage and maintain data. Automatic data transfer through interfaces is a way to reduce errors and improve the quality of the data. [25]

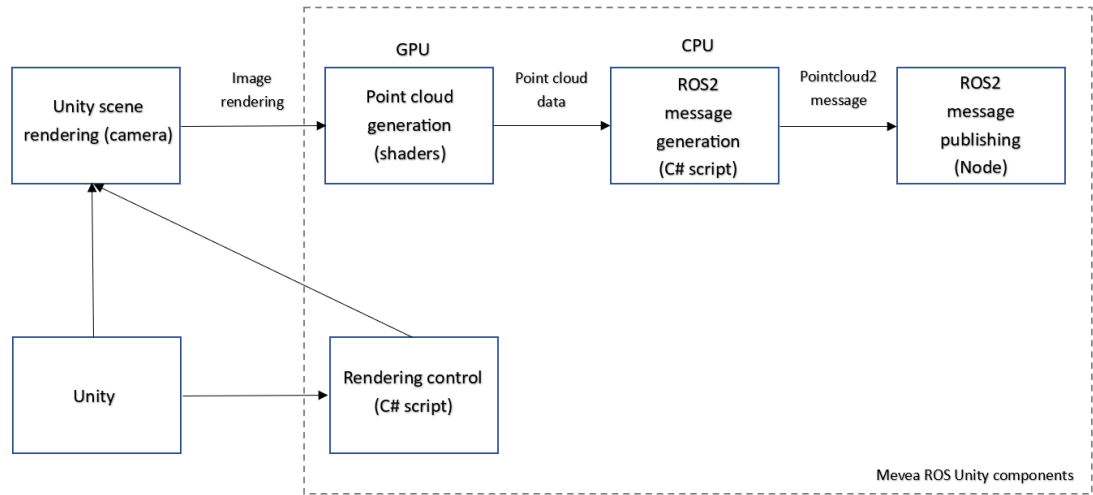
There is a Mevea API between Mevea and Unity that provides a generalized way to transfer simulation data from Mevea Solver to 3<sup>rd</sup> party software for visualization, data

collection and control. Thus, Mevea Api is a valid way to connect Mevea and Unity with each other. Mevea API is based on coding files written in C# language. These files make Mevea Solver act as a socket server that clients can connect to using Dynamic-link library (DLL). In this way, the location and orientation of the bodies can be read from Mevea and reproduced by Unity. This data transfer principle between Mevea and Unity is described in figure 9.



**Figure 9.** Mevea Solver simulation data to Unity. Based on [26].

When the simulation data from Mevea has been transferred to Unity, the next step is the integration between Unity and ROS2. Unity's camera component is like a depth camera and the Unity scene rendering can be used to generate point cloud data. Point cloud generation is done from image rendering using graphics processing unit (GPU) computing. The next step is to generate the point cloud data as a message in ROS2 format, so that the point cloud data can be visualized or utilized for algorithm development. ROS2 message generation is done using central processing unit (CPU). Publishing a ROS2 format message from Unity scene rendering is described in the flowchart in figure 10.



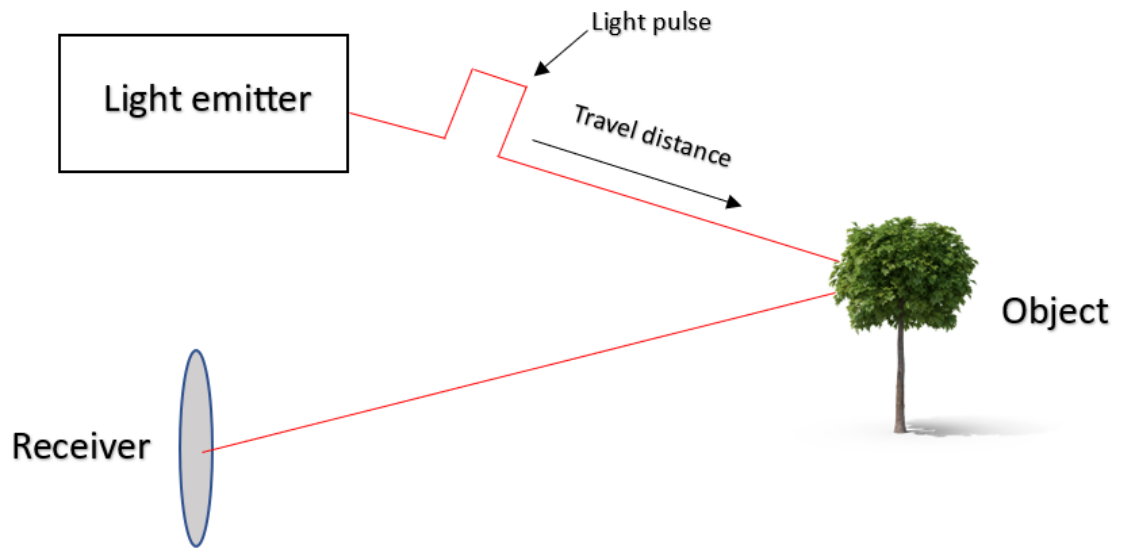
**Figure 10.** Flowchart of publishing ROS2 format message from Unity scene rendering.

## 4. LIDAR SIMULATION USE CASES WITH CHOSEN TOOLS

There are commonly used three types of sensors behind operator assistant systems: camera, radar and LiDAR. The differences between camera, radar and LiDAR are that cameras produce 2D images of the environment and radars and LiDARs can provide a 3D model of the environment, which is an advantage when accuracy is crucial. The underlying mechanism behind radar and LiDAR is similar but radar utilizes radio waves and LiDAR utilizes laser lights to detect surrounding objects. LiDARs can model up a very precise and high-resolution 3D depiction of the surveyed environment compared to radars. However, LiDARs are quite expensive and they require a large amount of computing power compared to radars and cameras. [27]

LiDAR provides the most comprehensive performance in various challenging conditions and the most accurate depiction of the environment compared to other sensors. Therefore, the LiDARs are the foundation of operator assistant systems and autonomous driving of forest machines. Cameras and radars provide redundancy for optimal detection and safety. Consequently, there will be a need for LiDAR simulations in the future and that is why LiDAR was chosen among the sensors for the use cases of this thesis.

LiDAR (Light Detection and Ranging) is a sensor technology that uses laser beams to measure distances and create a high-resolution 3D representation of the surveyed environment. LiDAR sensor emits invisible pulsed light waves into the surrounding environment and these pulses are reflected when they reach objects. Then the receiver picks up the reflected pulses and uses the speed of light and elapsed time to identify the locations of the objects. When this process is repeated millions of times per second, a precise real-time point cloud of the environment can be created. [27][28] The operating principle of the LiDAR is demonstrated in figure 11.



**Figure 11.** Operating principle of LiDARs. Based on [29].

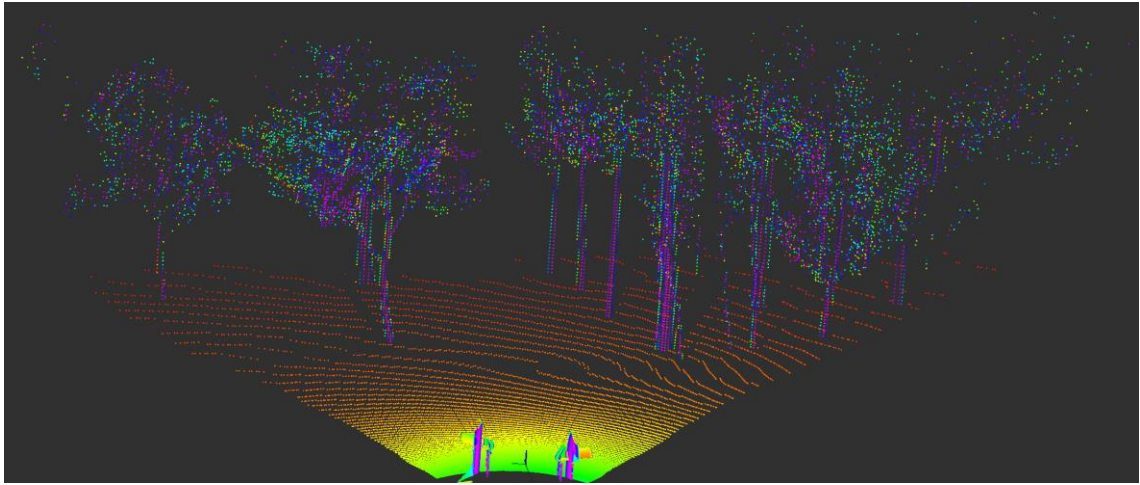
Ponsse has specific requirements for the LiDAR simulations and the suitability of Mevea's sensor simulation process for Ponsse's needs is evaluated with four different essential cases: the ability to detect trees, the ability to simulate different LiDAR types, the ability to change the location of the LiDARs and the effect of different locations on the simulation result and the ability to emulate challenging weather conditions and their effect on simulation result. For these cases, two virtual LiDARs were attached to the forest machine in Unity. Initially, the LiDARs were attached to the front frame and rear frame of the forest machine. Thus, an accurate depiction of the environment surrounding the machine can be produced. The goal is that these cases support the decision whether the Mevea's simulation process or some alternative software and process is suitable for Ponsse's needs.

#### 4.1 Tree detection

One of the most important cases in the simulations of forest machine operator assistant systems is that the virtual LiDAR detects trees and other relevant obstacles in the surrounding environment. There might be driving situations where it is enough if the sensors perceive obstacles roughly in the way of driving. However, high-resolution virtual LiDARs enable a more precise depiction of the surrounding trees and other obstacles.

The virtual LiDARs can provide a very accurate 3D representation of the environment in Mevea's sensor simulation process. Trees of different heights, shapes and thicknesses can be perceived from the visualized point cloud. In addition, the different altitudes of the

ground and all other obstacles, such as stones, bushes and logs are clearly visible. The point cloud visualization of the detected trees is presented in figure 12.



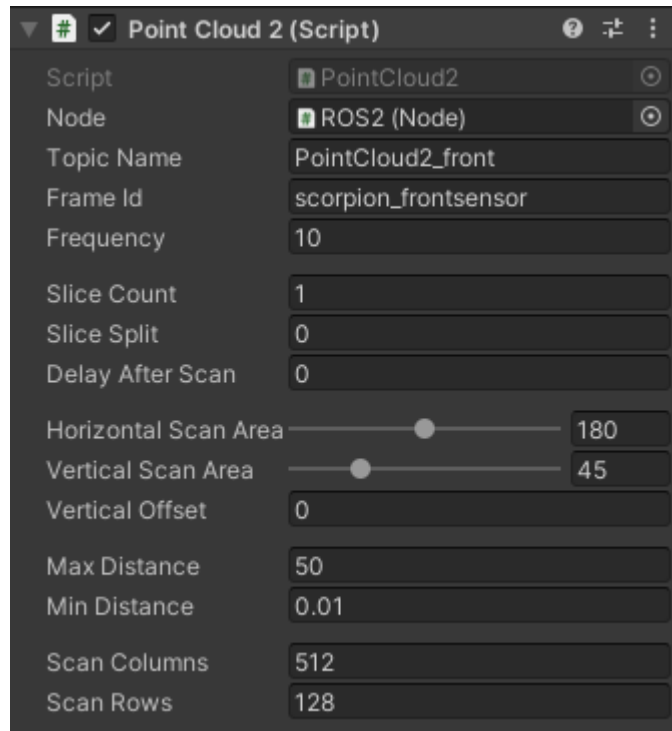
**Figure 12.** *A point cloud visualization of the detected trees.*

When the point cloud data provides accurate data about trees heights and circumferences, maps of forests and the operating environments can be created. Thus, it could theoretically be possible to measure the length of a single log or the straightness of a growing tree in forest with LiDAR.

## 4.2 LiDAR types

The modelling and simulation possibilities of different LiDAR types are important in sensor simulations. Imitating different properties of LiDARs in simulations supports choosing the right type of LiDAR for the real forest machine with lower risks and costs.

In Mevea's sensor simulation process, the virtual LiDARs defined in Unity are based on Unity's camera component, so there are no different LiDAR types to choose directly. Instead, the parameters of the virtual LiDAR can be edited in detail in the Unity Editor. All the adjustable parameters of the virtual LiDAR are presented in figure 13.



**Figure 13.** LiDAR parameters in Unity.

The LiDAR parameters to adjust are frequency, vertical and horizontal field of view, maximum and minimum detection distance and vertical and horizontal resolutions of the point cloud. Although different LiDAR types cannot be directly chosen, it is possible to emulate different LiDARs by manually adjusting these parameters. Therefore, this sensor simulation process supports finding the optimal LiDAR type for the forest machine's operating environment.

### 4.3 LiDAR locations

Simulations can also be used to evaluate and test the best locations for different sensors before installing and testing them on a real machine. The challenge with LiDAR placement is to find a location that reduces the dead zones around the forest machine and improves the point cloud resolution. A high-resolution LiDAR is quite expensive and installing several LiDARs on one forest machine becomes difficult. However, fewer LiDARs can create a larger dead zone where obstacles are not detected. Simulations can be used to estimate the necessary number of LiDARs and their optimal locations to perceive the surrounding environment.

In this sensor simulation process, ROS2 processes the data gathered from the simulation process and the visual process. Thus, ROS2 receives coordinate data and interprets the locations of the LiDARs attached to the forest machine continuously while the machine is driving. The locations of the LiDARs attached to the forest machine can be changed

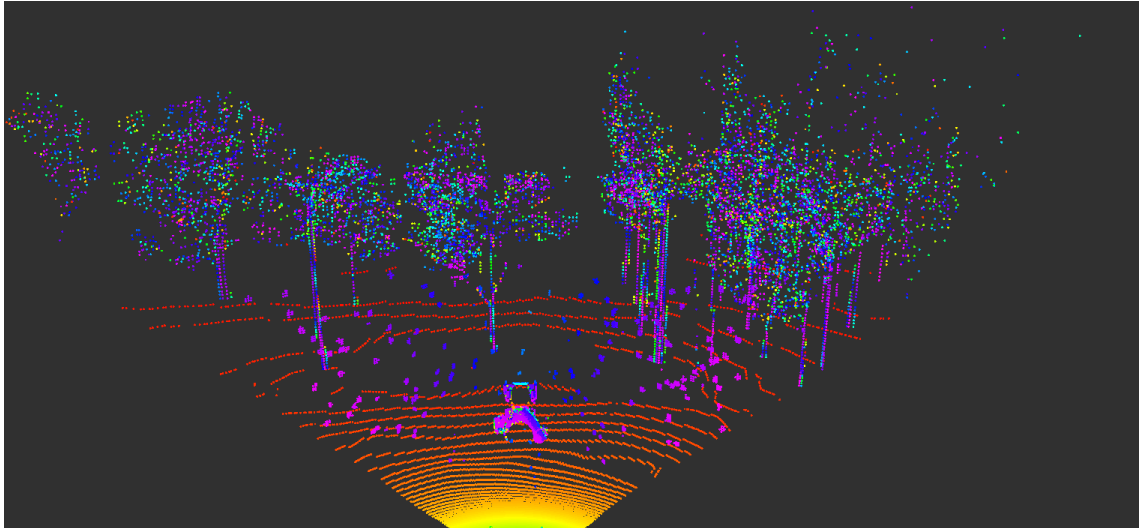
either in Mevea Modeller or Unity. Mevea Modeller is an option if a Unity license is not available and the Unity model is entirely created by Mevea. In this case, the LiDAR objects are dummies and the locations can be changed by moving the corresponding dummies in Mevea Modeller and the locations are updated automatically in Unity. If the Unity license is available, the locations of the LiDARs can be changed directly in Unity. Either way, the LiDAR objects can be relocated precisely to the desired locations and the suitability of the sensor locations can be easily evaluated using the point cloud visualized by RViz. Therefore, this sensor simulation process supports finding the optimal locations and number of LiDARs to minimize dead zones and maximize the density of points.

#### **4.4 The effect of weather conditions**

The ability to simulate different weather conditions is a very critical feature for sensor simulation software. Forest machines operate in challenging conditions which can affect both detection ability of LiDARs and operation of the forest machine. Challenging conditions must be considered in product development when designing operator assistant systems.

In this sensor simulation process, weather effects can be implemented either in Mevea or Unity. Weather conditions, such as rain, snowfall, wind and fog can be modified quite precisely in Mevea Modeller and Mevea Solver. However, these weather conditions only affect Solver visualization. Instead, the graphical weather conditions implemented in Unity are visible in Unity's visualization and thus also have an effect on the LiDARs point cloud.

Unity has a wide variety of different weather conditions and the possibility to change the time of the day, but these are only graphical effects. Thus, the weather conditions have no direct effect on the behavior of the simulated machine or the ability to detect obstacles with LiDARs. On the other hand, in LiDAR simulation cases, weather effects can be graphical effects that have only little effect on the detection ability of LiDARs because the real LiDAR also has its own light source, which enables good performance even in dark and challenging conditions. LiDARs are not dependent on an external light source as cameras are. For this reason, darkness or sunlight rays do not cause distortions in LiDARs perception, neither in this simulation process nor in reality. In this simulation process, the most significant effect of graphical weather effects is rain and snowfall which can be perceived as floating points in the visualization of the point cloud. The snowfall is demonstrated in the point cloud visualization in figure 14.



**Figure 14.** *Snowflakes in point cloud visualization.*

In sensor simulations, it would be essential to simulate challenging conditions where there is dirt on the sensor and the ability to detect obstacles is not accurate. There are no ready-made lens stains for emulating challenging conditions in this process. However, the LiDAR parameters can be manually modified in Unity to emulate challenging conditions. Manually emulated stains are not very accurate, so ready-made stains, mud or ice filters could produce a more realistic point clouds in challenging conditions.

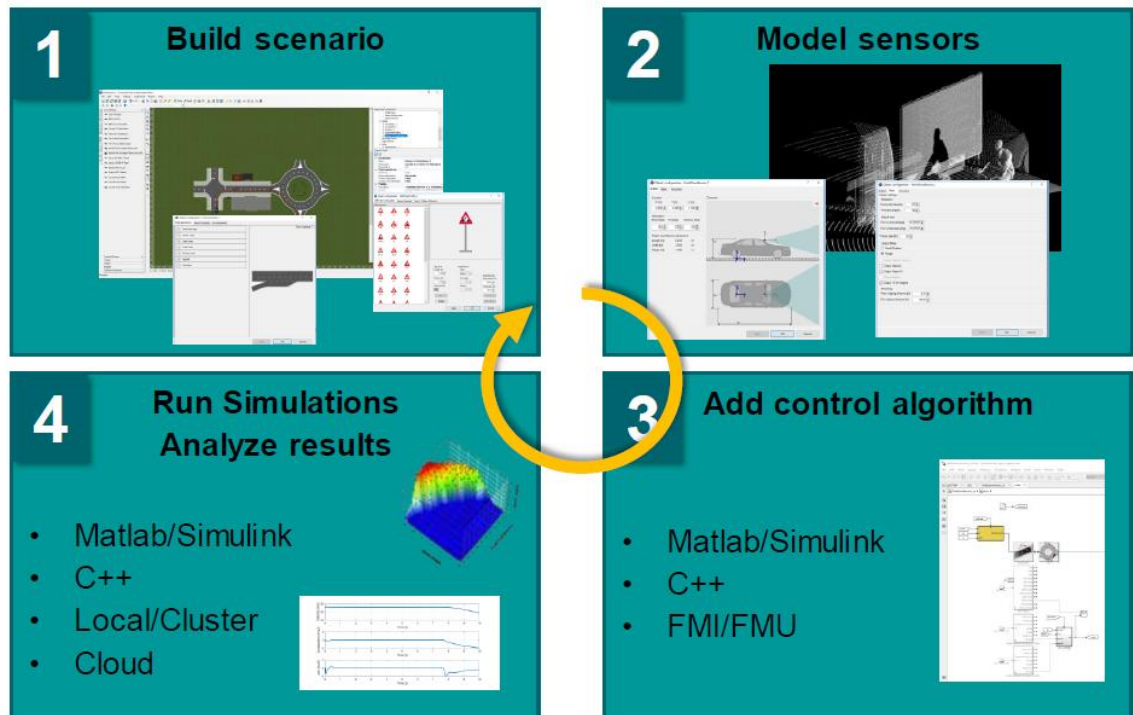
## 5. ALTERNATIVE SENSOR SIMULATION TOOLS AND PROCESSES

The sensor simulation process with Mevea Simulation Software, Unity and RO2 is only one of the possible simulation processes of operator assistant systems. There are countless amounts of different simulation software available for sensor simulations on the market nowadays. The alternative sensor simulation tools surveyed in this thesis are provided by Siemens, Creanex, Cognata and CARLA. The present chapter will introduce these alternative simulation tools and simulation processes and analyze their capabilities to meet Ponsse's requirements.

### 5.1 Siemens Prescan

The first alternative sensor simulation software is Siemens Prescan. Prescan is a physics-based simulation platform that is used for development of virtual verification for Advanced Driver Assistance Systems (ADAS) that are based on sensor technologies such as LiDAR, Radar and camera. Prescan focuses mainly on the key features found to be relevant by the automotive industry and the purpose of this thesis is to research if this software also meets the simulation requirements of the forest industry.

A simple sensor simulation combination was created to evaluate Prescan's capabilities to meet Ponsse's simulation requirements. Creating the simulation combination also clarifies the workflow of Prescan's simulation process. The basic idea of Prescan's simulation process is to create an experiment in the Graphical User Interface (GUI) and simulate the experiment using Simulink. Figure 15 presents the basic user workflow of Prescan.

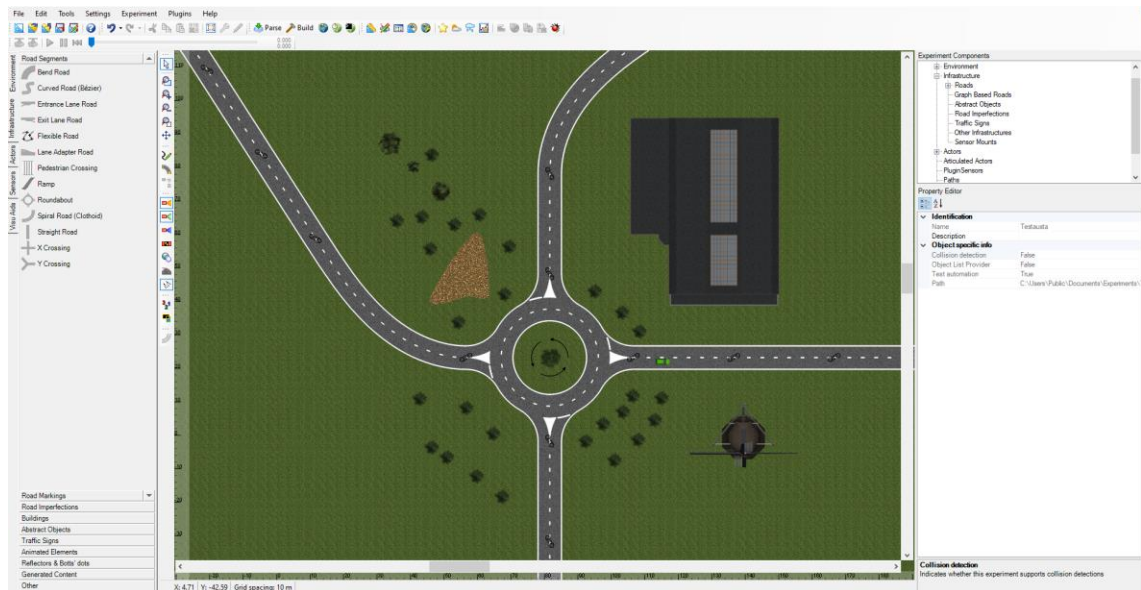


**Figure 15.** Prescan simulation process workflow [30].

To create an experiment, the first step is to build a scenario using the Prescan GUI. GUI offers a wide variety of environments, road sections, infrastructure components, actors, weather conditions and light sources. These predefined and freely configurable library elements enable the creation of powerful development and evaluation environment for intelligent vehicle systems.

The second step is to equip the vehicles of the created scenario with different sensor models. The selection of sensor models includes camera, laser, radar, LiDAR, and GPS. Sensor properties such as resolution, field of view and accuracy are adjustable. Thus, the suitability of different sensors for different purposes and conditions can be accurately evaluated.

The first and second steps of the basic user workflow are implemented in the Prescan GUI which is presented in figure 16.



**Figure 16. Prescan GUI.**

The third and fourth steps of the basic user workflow are implemented in Matlab and Simulink. A COM-based interface allows Prescan to communicate with Matlab and Simulink. Thus, the user can add control systems to design and verify algorithms for data processing and sensor fusion. The last step of the workflow is simulating the experiment and analyzing the results using Simulink.

Prescan has flexible interfaces to link with 3<sup>rd</sup> party vehicle dynamics model or 3<sup>rd</sup> party HIL simulators. In the research of this thesis, the dynamics of the forest machine was imported from Simcenter Amesim. The complex dynamics model of a forest machine requires a lot of customization to work properly on the Prescan platform. Also, importing forest terrain models into Prescan using Model Preparation Tool requires additional customization. Thus, building forest scenarios that meet Ponsse's requirements is not straightforward on Prescan.

Prescan's strengths are clearly in the development of autonomous systems for on-road vehicles. Prescan adds value when being used for concept studies where typical task is to evaluate different sensing systems or concepts or to evaluate different sensor fusion concepts. Therefore, sensor models and sensor simulations are advanced. Prescan has 2 rendering engines to visualize an environment. OpenSceneGraph is used in Normal Camera Sensor and Unreal Engine is used in Physics Based Camera Sensor. Prescan meets the requirements set for LiDARs and other sensors. For example, the generated point cloud data is accurate and the configuration options are sufficient to find suitable LiDAR types and locations to reduce the dead zones around the forest machine.

The other most significant advantages of Prescan are the configurable environmental conditions. The weather conditions can be adjusted to emulate the challenging operating conditions of the forest machine. In addition, the lighting settings can be emulated to match the lighting according to the day and time of the desired time zone. Prescan also offers the possibility to simulate driving lights. Thus, operating in the dark with the forest machine's work lights could be simulated.

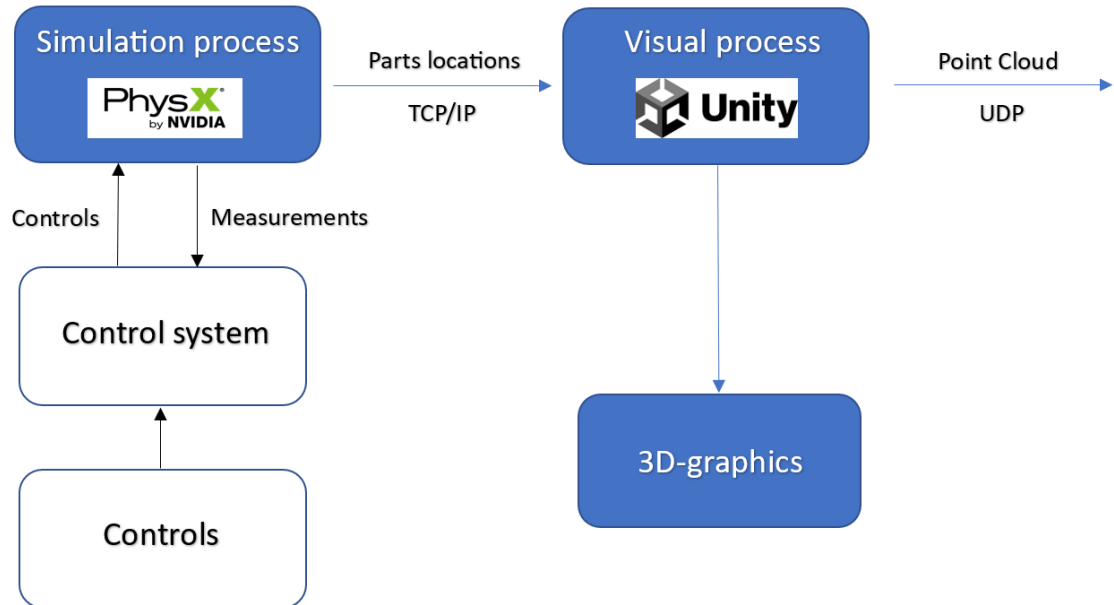
For Ponsse's future development, it is an advantage that Prescan can be used from MIL simulations to real-time test with SIL and HIL systems. Real-time simulation is a requirement for integration with test automation. Another good feature for future development is machine learning possibilities. Machine learning is not directly Prescan's field, but the communication connection with Simulink enables the implementation of machine learning solutions using sensor data.

Although Prescan offers a state-of-the-art simulation platform and tools for the development of autonomous vehicles, it does not provide enough benefits for Ponsse. One key requirement is to create harvesting scenarios on a simulation platform and Prescan does not meet this requirement properly. Creating relevant forest scenarios requires a lot of customization and grabbing or sawing trees is impossible on this platform. Therefore, the development and simulation of advanced operator assistant systems, such as a self-loading crane, is not possible with Prescan. Prescan brings benefits for Ponsse only in the field of driving and perception which are also significant when developing autonomous forest machines. However, it is critical in the forest machine operator assistant systems simulations that the simulation platform enables the autonomous development of other harvesting steps which requires the ability to cut and process trees with virtual forest machines.

## **5.2 Creanex**

The second alternative LiDAR simulation process is provided by Creanex. Creanex has implemented this simulation in a training simulator which has been developed to train forest machine operators. The basic process structure behind this LiDAR simulation solution is quite similar to Mevea's process. The simulation process and the visual process are separate processes. Forest machine's dynamic simulation model is created with Nvidia's PhysX physics engine and the visual process is based on Unity's graphics engine. Controls from the control system move the simulated machine and the locations of the machine parts are sent as a TCP/IP message to the visual process. In the visual process, the state of the simulation is visualized and the LiDAR point cloud based on 3D graphics is simulated. The visual process computes the simulated point cloud and sends

it forward as a UDP packet. The process diagram of the Creanex's LiDAR simulation process is described in figure 17.



**Figure 17.** LiDAR simulation process in Creanex's training simulator.

In Creanex's LiDAR simulation process, the kinematic structure of the forest machine is defined in a configuration file that is linked to the C++ code. The advantage of the configuration is that the Creanex library components can be utilized better in the implementation and it facilitates the C++ implementation of the simulation. The PhysX physics model is based on this kinematics configuration. Actors are created for the parts and the masses of the collision models are defined in the code. PhysX computes inertias for the actors based on the collision models and masses. Thus, the dynamics parameters are mainly in the code.

LiDAR parameters are also configurable. The simulated LiDAR object is attached to the desired part in the configuration file on the simulation side of the process. Therefore, the location of the LiDAR is updated for the visual process where the actual point cloud computation is done. A point cloud could also be generated from collision models using the PhysX engine's ray tracing method. However, the ray tracing method is quite heavy and limits the number of scanned points. When the visual process simulates the LiDAR point cloud, more scanned points can be generated and tree shapes and branches are more realistic in the point cloud representation.

The terrain models of the simulator use publicly available elevation data. The basic idea is that 3D environment models are different in visual simulation and dynamic simulation. The collision model is exported from the visual terrain model to the simulation process. Thus, dynamic simulation interacts with the environment.

Creanex's simulation process includes features that Ponsse requires from a sensor simulation. Creating scenarios in different forest environments is feasible and LiDAR parameters and locations can be easily modified. In addition, the point cloud generated from the visual process is realistic and accurate. The downside is that this training simulator solution does not include hydraulics. Thus, the user of the simulator does not control the hydraulic flow or valve openings but directly the movement of the boom. Therefore, it is not possible to determine, for example, the magnitude of the forces or the cylinder pressures. From Ponsse's point of view, this process would require more real physics and hydraulics in the future.

### **5.3 Cognata**

The third alternative sensor simulation software supplier is Cognata, an Israeli software company that provides a simulation platform for autonomous vehicle solutions for cities and unmarked terrains. Cognata's main focus is on the automotive industry, but the company also invests in the defense, agriculture, construction and mining industries. The key features of this platform are sensors, simple integrations and vehicle physics. The simulation combination was not created on this platform and the suitability for Ponsse's needs is cursory evaluated based on a few demo sessions by Cognata.

Cognata's simulation platform includes features that meet Ponsse's requirements. The user interface is impressive and creating forest environment scenarios is simple. 3D forest environment can be imported into the platform and additional selection of forest objects is versatile. Cognata also supports vehicle physics. There are more than a hundred vehicle physics parameters to adjust, including parameters related to tires, engine, steering, brakes and suspension. There is a selection of ready-made vehicles and the possibility to import customized vehicles into the platform.

This platform offers advanced sensor models. The platform includes many pre-parametrized sensors, such as a selection of Velodyne LiDARs and Ouster LiDARs. The parameters and locations of the sensors can be easily modified via the user interface. Realistic environmental challenges that affect detection can be emulated with advanced sensor filters, such as dust, mud and grime filters.

The created scenario is simulated with Cognata's physics engine and the visualization of the simulation process is based on the Unity game engine. All data gathered after the simulations can be analyzed with the tools provided by the Cognata platform. In this sensor simulation process, the generated point cloud data is not too ideal, as it can be in many cases. Due to various sensor filters, the point cloud data generated from the simulation environment can be closer to the real-world reality, especially in challenging conditions.

Cognata's integration possibilities also meet Ponsse's requirements. Integration with Simulink supports modelling, simulating and analyzing complex systems. The closed-loop simulation also enables the connection between the control system or sensor models and Cognata. Cognata also supports ROS integration.

Although Cognata meets many of Ponsse's requirements and the software has capabilities for sensor simulations and the future development of autonomous driving, there are also some downsides why this software may not yet be suitable for Ponsse's practical R&D work. Based on demos, it can be concluded that the physics of the forest machine would require a lot of customization to work properly on this platform. Also, tree harvesting needs could be difficult to implement with this software. In addition, the contents and prices of the available license packages do not meet Ponsse's requirements. For example, the license packages contain a lot of components and tools for sensor simulations of on-road vehicles that are irrelevant for Ponsse. Due to these reasons, Ponsse does not have the resources to utilize all the potential and highly advanced features of Cognata in simulations of operator assistant systems.

## **5.4 CARLA**

The last alternative sensor simulation tools surveyed in this thesis are provided by CARLA. CARLA is an open-source simulator for the development of autonomous driving systems. Its main focus is on the automotive industry and the key features are realistic urban environment simulations and sensor simulations. Due to its open-source nature it can also be used to develop autonomous forest machines in a forest environment.

This thesis focuses only cursory to the CARLA. The purpose is to survey the advantages and disadvantages of the CARLA simulator and its open-source environment from the perspective of Ponsse's requirements. Arttu Ruusiala has written a thesis on how a software only simulation environment could be created using available open-source components. The suitability of CARLA for Ponsse's needs is evaluated based on Ruusiala's thesis and discussions with the university about experiences with the CARLA simulator.

The sensor simulation combination was not created in this thesis and the feasibility of this simulation tool was therefore not tested or evaluated in practice.

CARLA's advantages are the possibilities offered by the open-source environment and the high-quality visual sensor simulations by the Unreal Engine game engine. The advantages of open-source software are reliability and flexibility. The reason why open-source software is reliable is that there are many skillful developers and users who make it possible to find bugs, fix them and continuously improve the quality. The open-source environment is also flexible because it is not limited to a specific patented IT architecture. Open-source software users can create a unique IT architecture that meets their requirements. In addition, the advantage of open-source software is its easy availability for free. By using open-source software instead of commercial software, tens of thousands of euros per year can be saved. [31]

The core of the CARLA simulator is a game engine, Unreal Engine. Unreal Engine is a powerful real-time 3D visualization tool with advanced rendering techniques and highly optimized code. However, one of the downsides is the video game physics. This simulation process focuses more on delivering high-quality rendering and sensor simulation instead of realistic vehicle physics [32]. When advanced physics is necessary, some other physics engine might be more suitable for Ponsse's needs. However, there is a possibility to integrate CARLA with several 3<sup>rd</sup> party applications to maximize its utility and extensibility. For example, Chrono is a multi-physics simulation engine providing realistic vehicle dynamics using templates. CARLA's Chrono integration allows adding Chrono templates to simulate vehicle dynamics. CARLA can also be integrated with the control system. Using ROS and Mathworks' ROS Toolbox, CARLA can be integrated with Simulink and MATLAB where autonomous driving and machine learning solutions can also be developed. Other Integration possibilities with 3<sup>rd</sup> party applications are also versatile and flexible via ROS bridge.

According to Ruusiala simulating challenging conditions is also CARLA's weakness. Most of the weather effects only affect the camera sensor but not the LiDAR data [32]. Thus, LiDAR simulation data might be too ideal compared to real data, as also noticed in the Mevea's use cases. Snowy and icy conditions are also missing [32]. Therefore, emulating the challenging operating environments of forest machines with software that focuses on urban environments and passenger cars could be tedious. Despite the customization, the sensors data might still be too ideal compared to reality.

The last disadvantage of open-source software is the lack of customer support. Commercial software suppliers support their users but in the open-source community no one

is responsible for helping users with their software issues. Sometimes issues are solved quickly in the community but there is no guarantee for this. [31]

## 6. DISCUSSION

This chapter focuses on discussing the suitability of the researched simulation tools and processes to Ponsse's R&D work. The feasibility of the simulation tools is evaluated based on the research done in this thesis and based on the needs of the future development of the forest industry.

Based on the survey, the visual process is one of the key elements in operator assistant system simulations. The two most popular rendering engines on the market are Unreal Engine and Unity. It cannot be directly said that one of the game engines is better, but the choice of game engine depends on the projects to be implemented and the skills of the user. Unreal engine is an open-source software and free to use for creating custom projects. The default features of an open-source game engine are often not as advanced as commercial game engines. However, due to the open-source code, the features of the game engine can be customized. Thus, users of open-source game engines need to be more experienced with game engines and coding.

Unity is a commercial game engine and the license is paid but quite reasonably priced for the company's R&D use. Unity does not offer as immersive simulation from the graphical point of view as Unreal Engine. On the other hand, Unity's other features make it easier to learn and use compared to Unreal Engine. Unity's ease of use is aided by the community. The community has solutions to various research and problem situations. In addition, there are numerous free tutorial videos on Unity tools that make project development easy even for more experienced developers.

Both game engines offer free and paid ready-made assets, such as various tree models, rocks, animals and other forest objects for creating 3D forest environments. The quality of the assets is better in Unreal Engine, but Unity Asset Store offers tens of thousands more assets than Unreal Marketplace. Thus, Unity's assets could offer better opportunities for creating diverse forest environments.

When creating the basis for the forest machine operator assistant systems simulations, Unity offers an easier and more straightforward way to build different forest scenarios. Unreal Engine would bring benefits compared to Unity only with better graphics and license price. However, Unity's graphics are good enough for simulations of operator assistant systems and the licenses are reasonably priced including customer support. Due to these reasons, Unity would be a worthy tool for the visual process of simulations of operator assistant systems.

The simulation combination requires a physics engine alongside the visual process, so that the simulations bring enough benefits for the development of operator assistant systems. The physics of the forest machine can be added to the simulation combination in two ways: The physics engine can be an external software, as in the sensor simulation process offered by Mevea or physics can be built-in the Unity ecosystem. Built-in physics engines available in Unity are Unity Physics, Nvidia PhysX and Havok. These physics engines provide a range of features such as rigid body dynamics and collision detection. The advantage of the built-in physics engine is that there is no third-party interface between the physics engine and the visual process. Thus, the interaction between the physical model and Unity's environment would be more realistic. As Creanex's sensor simulation process demonstrated, the downside of the built-in physics engine is the lack of real physics. One of the main requirements would be to include the real physics and hydraulics of the forest machine in the simulation process.

When the real physical properties are required, the Built-in physics could be replaced with AGX Dynamics, a multi-purpose physics engine developed by Algorix Simulation AB. AGX Dynamics for Unity is a Unity package which integrate AGX Dynamics into Unity and enables physics simulations inside Unity. This package contains significant modules for the development of virtual prototype of forest machine. For example, tire, drivetrain, hydraulics, terrain and cable modules combined get forest machine models to move as they should, using realistic parameters. [33]

Mevea's physics engine, which was analyzed in more detail in this thesis, offered significant advantages alongside Unity. Mevea is a flexible simulation tool that allows for multiple variations to find optimal solution. It is also capable of real-time simulations even with slightly more complex models. This contributes to the integration of test automation. If a virtual prototype were made with the tools provided by Mevea, the hydraulics of the forest machine might have to be simplified for simulations. A more suitable tool for hydraulic design might be, for example, Amesim with ready-made hydraulic components. Thus, it may be necessary to consider co-simulation between Amesim and Mevea when the importance of hydraulics increases in the simulations of operator assistant systems. Mevea I/O toolbox also supports connecting other external systems to the simulation. For example, a direct interface for Simulink enables integration with the control system. In addition to physics model tools, Mevea offers the opportunity to model and develop environments. Mevea can be used to create diverse forest environments and deformable terrains for simulations. Various challenging weather conditions typical of forest machine operating environments, such as rain, snow and fog, can also be added to the simulations. Mevea also enables interaction between the environment and the physical model.

The ability to grab and saw trees is a necessary feature when simulating operator assistant systems. However, it must be considered that if half of the environment is implemented in Mevea and half in Unity, the interaction of the modeled machine with environment and trees may be distorted due to the interface between simulation tools.

One of the main advantages of Mevea is the focus on simulations of off-road machines. With suitable simulation tools, it must be possible to develop and test new features of forest machines and for this purpose Mevea Simulation Software offers the possibility. There are numerous alternative commercial simulation platforms and physics engines on the market for the development of operator assistant systems and autonomous vehicles. However, most of these simulation platforms mainly focus on the development of autonomous systems of on-road vehicles in road traffic environments. These platforms do not sufficiently meet the requirements set for the simulation software by Ponsse.

The simulation platforms provided by Siemens Prescan, Cognata and Carla are more focused on the development of advanced driver assistance systems and autonomous on-road vehicles. This means that in addition to advanced perception tools and sensor models, these software only support simple physics for vehicles. The physics of the base machine that meets Ponsse's needs requires a lot of customization and the implementation is not straightforward on these platforms. In addition, implementing the movement or hydraulics of the boom would be challenging. These physics engines also do not support the interaction with trees and other forest environments. Sawing or grabbing trees is not possible on these platforms, which makes it impossible to simulate or develop the essential operator assistant systems. Siemens Prescan, Cognata and Carla are highly capable and advanced simulation platforms for the development of autonomous vehicles, but they contain many unnecessary features and components that are not useful for forest environments simulations. Due to these reasons, these alternative physics engines do not meet Ponsse's software requirements well enough and are therefore not suitable tools for practical R&D work.

The simulation process of the operator assistant systems that meets Ponsse's requirements requires a rendering engine and a physics engine. In addition, the key element of the simulation combination are the sensors that perceive the environment. LiDARs are the basis for systems that assist operators and automate harvesting phases. Thus, simulation combination must include tools for generating accurate LiDAR point cloud data.

One option for generating and visualizing point cloud data is the integration between Unity and ROS2 which was demonstrated in Mevea's sensor simulation process. This process utilizes GPU computing to generate point cloud data and CPU computing to

visualize it from Unity's ready-made camera component rendering. This process of creating point cloud data requires an understanding of ROS2 and how to integrate ROS2 into the Unity environment. The feasibility of this process to practical R&D work was evaluated with the use cases in chapter 4. In summary, the heights and shapes of the trees can be perceived. Also, shapes of the ground and obstacles in the environment can be detected even under challenging conditions. An accurate point cloud data like this can be utilized in the development of algorithms.

Another alternative method to generate point cloud data is Unity's own Simulation Pro sensor package. This reasonable priced add-on package enables sensors emulation in Unity virtual worlds to test system behavior in simulated environments. The sensor package includes a set of pre-build generic sensor types and partner-validated name-brand sensors but there is also the possibility to create and test custom sensors. In addition to sensors, the sensor package includes ready-made scripts for generating accurate point clouds for further development. Therefore, Unity's sensor package provides a simple and straightforward method to generate point cloud data even for more unexperienced developers.

It is also possible to get various open-source solutions for creating point clouds in Unity. The advantage of these solutions is easy availability and free testing possibilities, but the challenge is the lack of user manuals and customer support. In addition, most solutions require an understanding and knowledge of the ROS environment and the C# code language to enable virtual sensors to perceive the environment and to visualize simulation data and point clouds.

In all these point cloud generation methods presented above, the sensor models and parameterizations are implemented in Unity. Thus, the simulation results and point clouds generated by Unity's LiDAR solutions provide a good basis for estimating the number, locations and types of LiDARs in the final physical prototypes under different operating conditions.

An alternative way to connect sensor models to the simulation combination is to integrate them from another software into the Unity environment. One typical platform for modeling sensor models and setting properties is Simulink. Simulink models can be connected to the Unity environment using the ROS Toolbox, which enables co-simulations between Unity and Simulink.

As presented in this thesis, there are several variations of sensor simulation combinations consisting of game engine, physics engine and virtual sensors solutions. However,

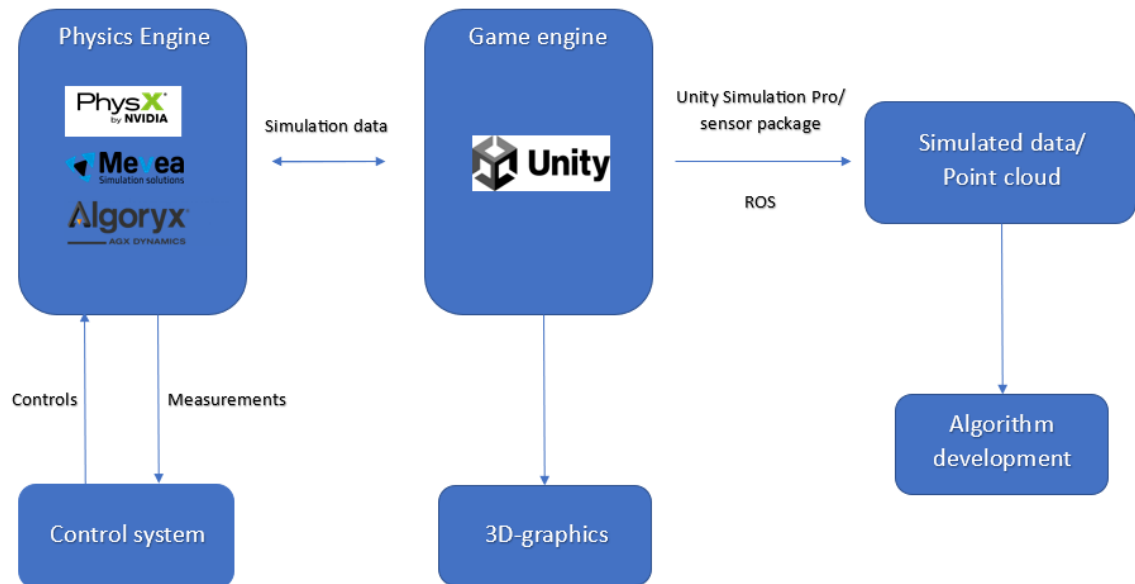
only a few of the variations available on the market were researched which were estimated to meet Ponsse's requirements based on the survey. Thus, there are countless amounts of other sensor simulation combination variations on the market for the development of autonomous systems which were not discussed in this thesis. The capabilities of the simulation software and simulation processes studied in this thesis to meet the sensor simulation requirements set by Ponsse are listed in Table 1.

**Table 1.** Summary of requirements and evaluation of surveyed sensor simulation tools

Requirements	Priority	Mevea (Unity)	Siemens Prescan (Unreal Engine)	Cognata (Unity)	Carla (Unreal Engine)	Creanex (Unity + PhysX)
Ease to build harvesting scenarios in a forest environment	HIGH	+	-	-	-	+
Integration to control system (Simulink, ROS etc.)	HIGH	+	+	+	+	+
Vehicle and boom physics (especially hydraulics)	HIGH	+	-	-	-	-
Forest environment modelling/maps	HIGH	+	-/+	+	-/+	+
Lidar, camera, radar simulation in virtual environment (Easy to position, test different sensor types and create challenging conditions)	HIGH	+	+	+	+	+
Accurate Point cloud data and possibility to change resolution	HIGH	+	+	+	-/+	+
Tree processing (harvesting, saw, load etc.)	HIGH	+	-	-	-	+
Machine learning capabilities	MEDIUM	+	+	+	+	+
Collision Avoidance possibilities (e.g. human detecting)	MEDIUM	+	+	+	+	+
Autonomous driving possibilities (path following & self-control)	MEDIUM	+	+	+	+	+
Customer support	MEDIUM	+	+	+	-	+
Price	MEDIUM	+	+	-	+	+

Based on the studies, the Unity environment with a built-in or external physics engine offers the most advantages for Ponsse's needs. Physics can be integrated into the simulation process using the external Mevea physics engine or using the built-in PhysX or Algorix physics engines. Then, there are a couple of options for generating LiDAR point

cloud data from the Unity environment and the most straightforward and thus most worthy solution is sensor implementation of Unity's own Simulation Pro sensor package. The final simulation process workflow and software options are presented in the process diagram in figure 18.



**Figure 18.** Final proposal for sensor simulation process and software options.

## 6.1 Further development

Once the simulation tools that meet the requirements have been found, it is necessary to outline how these can be utilized in practical product development work. The purpose of this chapter is to provide some guidelines on how the presented simulation process and simulation tools could contribute to the development of operator assistant systems and autonomous forest machines.

The simulation process where Mevea's physics engine is integrated with Unity's game engine could enable simulation-based design, development and testing of virtual prototypes in the simulator. Unity provides some potentially useful tools for future development of operator assistant systems and autonomous solutions in the Unity environment. In the future, Unity's add-on simulation tools could be utilized especially in the simulations and development of systems related to measuring, computer vision and machine learning.

Unity Asset Store offers paid measurement tools that could be utilized in simulations for tree size measurement, such as measuring tree lengths or circumferences using LiDAR. The measurement tools could also support the development of a system that measures



environments, it is possible to generate training examples in a customizable and controllable way, and the challenges of gathering and annotating data in the real world can be avoided. Capturing data from the simulations is easy and does not require as much manual labor as gathering data from the real world. Thus, simulations make this process easier.

Unity provides a realistic simulation environment for the development and testing of intelligent systems in a wide range of scenarios. An open-source add-on called The Unity Machine Learning Agents Toolkit (ML-Agents) could be utilized to train intelligent agents. Machine learning allows agents to learn automatically through reinforcement learning. The basic idea is not to tell the agents what to do, but to develop them to determine how to proceed with certain tasks. In the future, Unity's machine learning capabilities could be utilized, for example, to train artificial intelligence to autonomously load randomly placed logs. Thus, Unity Perception Package and ML-Agents together can be utilized in the development of operator assistant systems and in the optimization of energy efficiency.

This thesis mainly focuses on Model-in-the-loop and Software-in-the-loop simulation solutions which are conducted in the early stages of the software development process. Therefore, further research is suggested on how these simulation tools and simulation combinations could be utilized in the Hardware-in-the-loop simulations and how the proposed simulation combination could be integrated with test automation.

## 7. CONCLUSIONS

The complexity of autonomous forest machines increases exponentially compared to traditional ones. An integrated toolkit that supports the full development V-cycle is essential to enable cost and time efficient development processes. The aim of this thesis was to survey and research the suitability of different simulation tools and processes for the simulations of forest machine operator assistant systems. The surveying demonstrates that the simulation process workflow of forest machine operator assistant systems follows a very similar basic structure regardless of the chosen software and simulation tools. The key elements of the simulation process are the physics engine and the game engine. A suitable physics engine and game engine together with the sensor models enable the utilization of simulations in the development of autonomous systems.

In this thesis, several alternative variations of the sensor simulation processes were surveyed and the sensor simulation combination built around the Unity game engine proved to be the most potential and beneficial process. Unity offers a suitable platform and tools for creating harvesting scenarios in forest environments and for developing and testing the autonomous features of future forest machines. Sensors such as LiDARs and cameras are the foundation of operator assistant systems and autonomous forest machines. Unity supports the modelling of sensors or integrating sensors from other software into the Unity environment. Therefore, simulating sensors and generating point clouds from the Unity environment is straightforward and Unity's sensor simulation process enables placement optimization of multiple sensors for autonomous forest machines and future development.

When the physics engine, such as Mevea or Algorix, is integrated into Unity, the physics of the forest machine can be included in the simulation process. This enables vehicle and boom physics simulations and supports the development and testing of new features of forest machines. These physics engines also enable realistic contact with the environment. Thus, simulations can be utilized, for example, to develop autonomous tree loading or processing systems.

In this thesis, sensor simulation software focused on the automotive industry were also surveyed to understand the strengths and weaknesses of different software intended for the development of autonomous systems for on-road vehicles. These platforms only provide development tools related to driving and detection. The possibility to process trees, which is one of the most important requirements for simulations of operator assistant

systems in forest machines, is not feasible on these simulation platforms. Therefore, the capabilities of software focused on the automotive industry do not meet the needs of the forest industry.

## REFERENCES

- [1] P. Seppälä, Tavaralajimenetelmän metsäkoneiden automaation kehitysnäkymät, Metsäteho Oy, Vantaa, Dec 2020. Available (accessed Jan 26, 2023): <https://www.metsateho.fi/wp-content/uploads/Raportti-259-Tavaralajimenetelman-metsakoneiden-automaation.pdf>
- [2] J. Mattila, Introduction to Model-Based Design. Model-Based and Rapid Prototyping lecture 1, Tampere University, Jan 2022.
- [3] M.Ahmadian, Z. Nazari, N. Nakhaee, Z. Kostic, Model Based Design and SDR, Sep 2005, pp. 1-6. Available (accessed Jan 31, 2023): <https://ieeexplore.ieee.org/document/1575352/metrics#metrics>
- [4] Collimator, Model Based Design Overview for System Development, Nov 2022. Available (accessed Feb 1, 2023): <https://www.collimator.ai/post/model-based-development>
- [5] A. Bergmann, Benefits and Drawbacks of Model-based Design, Sep 2014, pp. 15-19. Available (accessed Feb 1, 2023): [https://www.researchgate.net/figure/Model-based-Design-Workflow-The-MathWorks\\_fig1\\_270494068](https://www.researchgate.net/figure/Model-based-Design-Workflow-The-MathWorks_fig1_270494068)
- [6] J.Banks, Handbook of simulation: principles, methodology, advances, applications, and practice Principles of Simulation, 1998, pp. 3-29. Available (accessed Jan 25, 2023): [https://app-knovel-com.libproxy.tuni.fi/web/view/khtml/show.v/rcid:kpH-SPMAAP9/cid:kt003Z4236/viewerType:khtml//root\\_slug:handbook-simulation-principles/url\\_slug:part-i-principles?b-toc-cid=kpHSPMAAP9&b-toc-root-slug=handbook-simulation-principles&b-toc-title=Handbook%20of%20Simulation%20-%20Principles%2C%20Methodology%2C%20Advances%2C%20Applications%2C%20and%20Practice&b-toc-url-slug=part-i-principles&kpromoter=federation&view=collapsed&zoom=1&page=1](https://app-knovel-com.libproxy.tuni.fi/web/view/khtml/show.v/rcid:kpH-SPMAAP9/cid:kt003Z4236/viewerType:khtml//root_slug:handbook-simulation-principles/url_slug:part-i-principles?b-toc-cid=kpHSPMAAP9&b-toc-root-slug=handbook-simulation-principles&b-toc-title=Handbook%20of%20Simulation%20-%20Principles%2C%20Methodology%2C%20Advances%2C%20Applications%2C%20and%20Practice&b-toc-url-slug=part-i-principles&kpromoter=federation&view=collapsed&zoom=1&page=1)
- [7] P.Cantot, D.Luzeaux, Simulation and Modeling of Systems of Systems, 2011, pp. 5-12. Available (accessed Feb 2, 2023): <https://ebookcentral.proquest.com/lib/tampere/reader.action?docID=1143608>
- [8] A.M.Law, Simulation Modeling and Analysis, 5<sup>th</sup> edition, Tucson, Arizona, USA, 2014, pp. 71-72. Available (accessed Feb 2, 2023): <https://industri.fatek.unpatti.ac.id/wp-content/uploads/2019/03/108-Simulation-Modeling-and-Analysis-Averill-M.-Law-Edisi-5-2014.pdf>

- [9] T.Nakkasem, V-Model: An improvement of Waterfall, Apr 2020. Available (accessed Mar 27, 2023): <https://medium.com/software-engineering-kmitl/v-model-3a71622b3d82>
- [10] JavaTpoint, V-model. Available (accessed Mar 27, 2023): <https://www.javatpoint.com/software-engineering-v-model>
- [11] Acsysteme, Model-Based Design. Available (accessed May 10, 2023): <https://www.acsysteme.com/en/customised-service/model-based-design/>
- [12] OPAL-RT Technologies, Software-in-the-loop. Available (accessed May 11, 2023): <https://www.opal-rt.com/software-in-the-loop/>
- [13] Mevea, Success Stories. Available (accessed Mar 17, 2023): <https://mevea.com/success-stories/>
- [14] Mevea Software for real-time simulation. Available (accessed Apr 4, 2023): <https://mevea.com/solutions/software/>
- [15] Mevea Ltd, Mevea Modeller: Beginner tutorials, version 4.1.3.
- [16] D.Tyler, How to Choose the Best Video Game Engine, 2021. Available (accessed Apr 12, 2023): <https://www.gamedesigning.org/career/video-game-engines/>
- [17] O.Elmofty, ROS2 vs. ROS1— key differences and which one is better?, Aug 2022. Available (accessed Apr 17, 2023): <https://medium.com/@oelmofty/ros2-how-is-it-better-than-ros1-881632e1979a>
- [18] ROS 2 Documentation: Humble, Understanding nodes. Available (accessed Apr 17, 2023): <https://docs.ros.org/en/humble/Tutorials/Beginner-CLI-Tools/Understanding-ROS2-Nodes/Understanding-ROS2-Nodes.html>
- [19] ROS 2 Documentation: Humble, Understanding topics. Available (accessed Apr 18, 2023): <https://docs.ros.org/en/humble/Tutorials/Beginner-CLI-Tools/Understanding-ROS2-Topics/Understanding-ROS2-Topics.html>
- [20] ROS 2 Documentation: Foxy, Understanding services. Available (accessed Apr 18, 2023): <https://docs.ros.org/en/foxy/Tutorials/Beginner-CLI-Tools/Understanding-ROS2-Services/Understanding-ROS2-Services.html>
- [21] ROS 2 Documentation: Foxy, Understanding actions. Available (accessed Apr 18, 2023): <https://docs.ros.org/en/foxy/Tutorials/Beginner-CLI-Tools/Understanding-ROS2-Actions/Understanding-ROS2-Actions.html>
- [22] ROS.org, Client Libraries. Available (accessed Apr 21, 2023): <http://wiki.ros.org/Client%20Libraries>
- [23] Automatic Addison, What is the difference between rviz and Gazebo?, Jun 2020. Available (accessed Apr 18, 2023): <https://automaticaddison.com/what-is-the-difference-between-rviz-and-gazebo/>

- [24] S.Doggett, What Are Point Clouds, And How Are They Used?, Dronegenuity. Available (accessed Apr 20, 2023): <https://www.dronegenuity.com/point-clouds/>
- [25] DNA, Ohjelmointirajapinta toimii siltana ohjelmistojen välillä, Jun 2022. Available (accessed Apr 14, 2023): <https://www.dna.fi/yrityksille/blogi/-/blogs/ohjelmointirajapinta-toimii-siltana-ohjelmistojen-valilla>
- [26] Mevea Ltd, Mevea ROS 2 Interface, Tutorial.
- [27] AUTOCRYPT, Camera, Radar and LiDAR: A Comparison of the Three Types of Sensors and Their Limitations, Aug 2021. (accessed Apr 26, 2023): <https://autocrypt.io/camera-radar-lidar-comparison-three-types-of-sensors/>
- [28] Velodyne Lidar, What is lidar? Available (accessed Apr 27, 2023): <https://velodynelidar.com/what-is-lidar/>
- [29] Frydlewicz Consulting for 3D-Vision, LiDAR and ToF Cameras – Technologies explained. Available (accessed Apr 24, 2023): <https://tof-insights.com/time-of-flight-lidar-and-scanners-technologies-explained/>
- [30] Siemens, Simcenter Prescan Training, Prescan GUI, 2021.
- [31] Cyber Writes Team, Pros and cons of using open-source software, Cyber Security News, Nov 2022. Available (accessed Jun 7, 2023): <https://cybersecuritynews.com/pros-and-cons-of-using-open-source-software/>
- [32] A. Ruusiala, Vehicle automation software development using software-only simulation, MA thesis, Tampere University, 2022.
- [33] Algoryx, Heavy Vehicles. Available (accessed Aug 14, 2023): <https://www.algoryx.se/heavy-vehicles/>