

LAPPEENRANTA UNIVERSITY OF TECHNOLOGY
LUT School of Energy Systems
LUT Mechanical Engineering

Ilya Kurinov

**DEVELOPMENT OF VIRTUAL REALITY USER INTERFACE FOR HEAVY
MACHINERY REAL-TIME SIMULATIONS**

17.09.2018

Examiner(s): Professor Aki Mikkola

D. Sc. (Tech.) Kimmo Kerkkänen

ABSTRACT

Lappeenranta University of Technology
LUT School of Energy Systems
LUT Mechanical Engineering

Ilya Kurinov

Development of virtual reality user interface for heavy machinery real-time simulation

Master's thesis

2018

68 pages, 29 figures and 9 tables

Examiner(s): Professor Aki Mikkola

D. Sc. (Tech.) Kimmo Kerkkänen

Keywords: Multibody System Dynamics, Real-Time Simulation, User Interface, Virtual Reality.

The traditional product development process is time consuming and requires a high amount of investments. The reasons for this are high amount of iterations during the design stage and need in building multiple prototypes before launching a production. Nevertheless, the resulting product might not fulfill all customer's needs during exploitation. Therefore, this thesis was concentrating on defining of a method, which allows to effectively collect the user preferences and use them in the development process.

The method reviewed in this thesis applies a Mevea real-time simulation software and UE4(Unreal Engine 4) game engine, which allow the user to test a machine model and change its physical parameters via VR(Virtual Reality) interface. For the development of the physical model of the machine was developed in Mevea software, and for development of the visualization and the user interface of the software was used UE4. The physical parametrization of the model was achieved by application of the Mevea XML file with initial parameters of the model and XML modification library in UE4. The obtained model parameters are filtered inside of the user interface according to the customers group.

The resulting software provides an opportunity to collect the user preferences automatically and use them in the product development process. This kind of software might reduce the amount of time and money invested into the development process without losing the performance, reliability and usability.

ACKNOWLEDGEMENTS

I want to say words of appreciation for family and friends, who have been supported and inspired me during my studies. Without their help and guidance, I would not succeed in this kind of challenging project.

The special thanks addressed to professor Aki Mikkola for providing an opportunity to work on this project and Dr. Kimmo Kerkkänen for thorough supervision of my work.

TABLE OF CONTENTS

ABSTRACT	1
ACKNOWLEDGEMENTS	2
TABLE OF CONTENTS	4
LIST OF SYMBOLS AND ABBREVIATIONS	6
1 INTRODUCTION	7
1.1 Research objective and questions	7
2 THEORY	10
2.1 Theory of crowdsourcing	10
2.2 The development process of user interface.....	11
2.3 XML overview	11
2.3.1 Structure of the simulation input XML document.....	13
2.3.2 Parsing and connection to the interface	18
2.4 Requirements for User interface	18
2.4.1 Communicating with the user through crafting menu	18
2.4.2 Comfort level of the view	19
2.4.3 Visualization	21
2.4.4 Interaction	21
2.4.5 Security requirements	22
2.4.6 Loading and running the simulation	23
2.4.7 Saving user-specified data	24
2.4.8 Providing tutorials and additional information.....	25
2.4.9 Control of other parameters: graphics, controls and appearance.....	26
2.4.10 Hardware, software and system requirements	26
3 RESULTS	31
3.1 Operation of the user interface.....	31
3.1.1 Application performance	31
3.1.2 Data input method.....	32
3.1.3 Main menu operation	35
3.1.4 Simulation operation.....	38
3.2 Building hardware for the user interface	51

3.3	Management data obtained	52
3.4	The main menu obtained data	53
3.4.1	User information data	53
3.4.2	Machine initial parameters.....	54
3.5	Data obtained during the simulation	54
3.6	Data obtained after the simulation	55
3.7	Output files structure	56
3.8	Data sending	58
3.9	Data filtering	60
4	ANALYSIS	63
4.1	Practical application example	64
4.2	Test group results.....	64
4.3	Achievement of research results	64
5	CONCLUSION	65
	LIST OF REFERENCES.....	67
	APPENDIX	

Appendix I: Main output CSV file

LIST OF SYMBOLS AND ABBREVIATIONS

VAL_{p1} – Value of parameter 1

X_a – Position of actor along X-axis [m]

X_m – Position of mesh along X-axis [m]

AR – Augmented Reality

HMD - Head Mounted Display

RAM - Random Access Memory

SGML - Standard Generalized Markup Language

UDP - User Datagram Protocol

UE4 - Unreal Engine 4

VR – Virtual Reality

XML- Extensible Markup Language

1 INTRODUCTION

The customer's product with the product correlates to the fulfillment of the requirements of the customer's needs [1]. The common practice in the industry is to involve the customer in the product development process in order to increase the customer's satisfaction rate. Accordingly, it is crucial to effectively share information between all parties such as customers, company and other internal groups involved in the development process.

The early stages of the product development are challenging due to a large number of professionals and customers from different backgrounds involved. Customers and industry oriented technical staff are normally not profoundly familiar with highly specific data provided by product development researchers. This data is typically obtained using mathematical model based simulations. This may lead to problems in communication and knowledge sharing between different individuals and teams involved in the product development process.

Recent studies related to the product development are showing the interest in the information sharing among the community. The researches in this field state that knowledge and information sharing is highly contributing to the speed and quality of the development of product innovations [2]. Most of the papers in this field are stating that there is a need for the development of the highly effective information transfer method inside the community [1], which will support to increase the effectiveness of a product development procedure.

1.1 Research objective and questions

The primary objective of this thesis is to create an efficient method for gathering user feedback from a customer and transforming it into the data, which is valuable for research and development groups in a company. This method should cover the high amount of respondent for gathering a reliable statistical sample. According to these assumptions, the method should be based on the digital solution, which applies real-time simulation with the game-like environment.

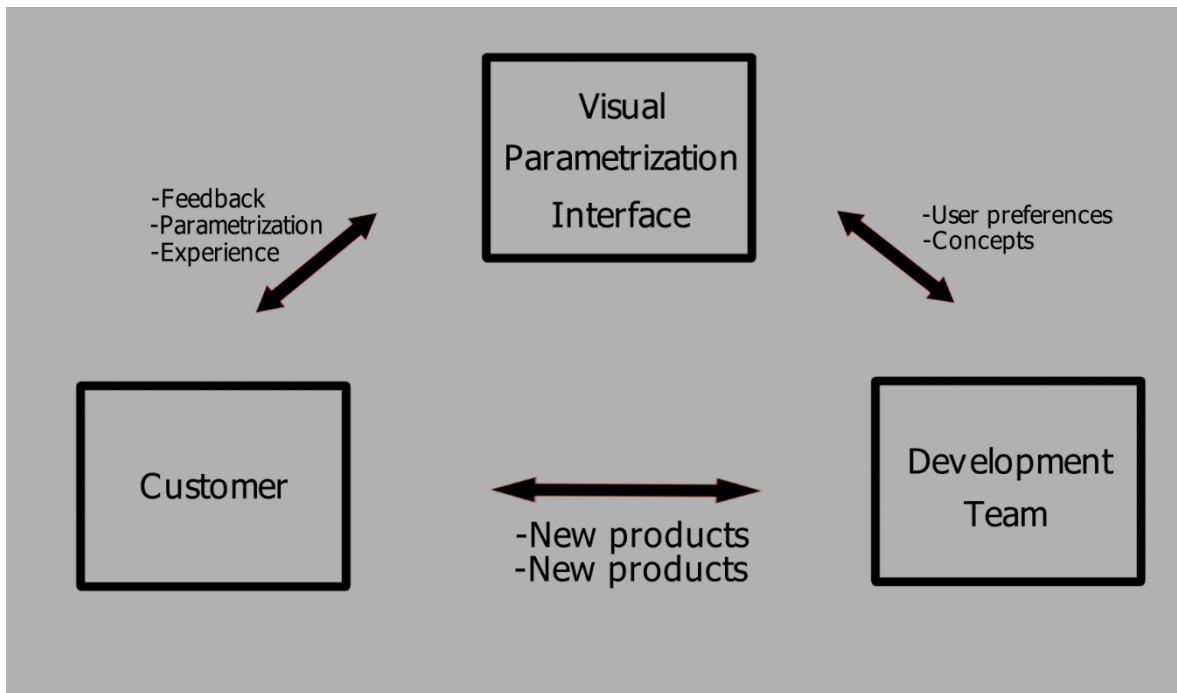


Figure 1: Objective for interaction between customer and development team

This thesis is concentrating on the improvement of interaction between user and company. The improvement can be achieved by the introduction of the interface shown in figure 1. The interface is collecting user feedback about desired physical parameters and experience, which are collected and sent to the development team. As an outcome of this data analysis, the development team creates new concepts and products. For achieving the goal of the study following research question must be studied:

- How can be made a simulation model parametrization? What kind of tools are required?
- How can be performed the simulation model visualization?
- How should be collected and analyzed a user-specified data? What is a method of converting user specified data into the concepts?



Figure 2: VR based interface

For the research was chosen a VR based interface based on the Oculus Rift virtual reality headset and Leap Motion controller hardware in Unreal Engine 4, which is shown in figure 2. This setup will give an opportunity to choose parameters based on the visual and simple technical aspects of the model without applying special software or modifying XML files by users.

2 THEORY

The task of the development of such interface includes aspects of several areas. For proper design of the application must be considered software engineering, crowdsourcing and a physical model of the machine issues. In the particular case, goals of the software development and physical modeling process are dependent on a crowdsourcing task. The crowdsourcing task specifies what kind of actions needed from the crowd. This task is created by the company, which needs to gather information from the crowd.

2.1 Theory of crowdsourcing

According to Daren, crowdsourcing is a type of participative online activity in which an individual, an institution, a non-profit organization, or company proposes to a group of individuals of varying knowledge, heterogeneity, and number, via a flexible open call, the voluntary undertaking of a task. There are four main types of crowdsourcing: knowledge discovery and management, broadcast search, peer-vetted creative production and distributed human intelligence tasking. In the knowledge discovery and management, the crowd got an assignment to collect information for a company. The second type of crowdsourcing is broadcast search, which implies giving to a crowd task to solve empirical problems, for example, scientific problems. The third type, peer-vetted creative production, concentrates on creating and selecting creative ideas. The fourth type of crowdsourcing is distributed human intelligence tasking, which implies giving to a crowd task to analyze a large amount of data. The tasks for a crowd can be ranging in complexity and needed resources. Depending on the case, the crowd is using work, money, knowledge and experience for completing the task. The key parts of crowdsourcing are an interested organization, a community that is completing a task, an online environment and mutual benefit for both parties. [3]

In the current setting, the interested organizations are companies, which are designing and producing heavy machinery. The community, which performs a design task voluntarily must be companies and operators of target machines. There are clear benefits for both parties involved in crowdsourcing. For interested companies, crowdsourcing will help to reduce the design budget and create more targeted products. The community has benefits of testing new

products, affecting the design process so, that end product fulfills their needs and reduction of a price without losing the quality. Therefore, there is a need for designing an online environment, which allows involved parties to interact with each other. [3]

2.2 The development process of user interface

A starting point of the development process is consideration of crowdsourcing type. In the case of this study is applied peer-vetted creative production. At the peer-vetted creative production organization tasks community with creating and selecting ideas. Therefore, the community must define the most suitable configuration according to their experience with the application. From this statement is derived a requirement for the application: the user interface must collect data via the visual interface, store and send configurations defined by community. [4]

The next stage is to define basic features of the application, called the high-level design of the application. High-level design includes decisions about the target platform, user interface and external interfaces used by the application. After defining the requirements for application, a top-down approach can be applied. This method implies dividing of the application functionalities into parts. These parts are divided into more detailed parts. Then, these parts are used to create needed classes and methods, which are needed for the program functioning. In the application created during this research, the target platform is Windows 10. The real-time simulation is performed by Mevea software package. For the creating main functionality of the model configuration used simulation XML files, which allows modifying initial parameters of the model.

2.3 XML overview

According to Joshi, XML stands for Extensible Markup Language and is a markup language used to describe data. It was developed by World Wide Web Consortium's (W3C) XML Special Interest Group in 1996 and was based on the SGML. Initially, it was developed for the application on the Internet, had design goals which are stated at the W3C Recommendations of XML 1.0 and associated standards. The design goals were concentrating on the efficiency and ergonomics with the minimal importance of the terseness of the language. According to that specification was created an extendable form of the

SGML, Standard Generalized Markup Language, which gives the user a possibility to define own structure of the document based on the XML production rules. [5]

The production rules of an XML document are concentrating on the logical and physical structure of the document. The logical structure is related to the formal grammar of a document. A document, which is meeting these rules called well-formed. [5] Comparison between well-formed and not well-formed document structure shown in figure 3.

Well-formed XML file	Not well-formed XML file
<pre><?xml version="1.0" encoding="UTF-8"?> <RootElement> <Element1>This is the well formed document</Element1> <Element2 Attribute1="1.0" Attribute2="Data"/> <Element3/> </RootElement></pre>	<pre><?xml version="1.0" encoding="UTF-8"?> <RootElement> <Element1>This is the invalid document <Element2 Attribute="1.0" Attribute="Data"> <Element3> </RootElement> </element1></pre>
<ul style="list-style-type: none"> - Document has corresponding end tag for every start tag - Document has proper elements nesting. Example: <Root><Element1></Element1></Root> - Element2 has different names for attributes: Attribute1, Attribute2 	<ul style="list-style-type: none"> - Element1 missing corresponding closing tag (case sensitive) - Document has improper elements nesting. Example: <Root><Element1></Root></Element1> - Element2 has same names for two attributes - Element3 is empty, but not marked as empty(<Element3/>) or closed (</Element3>)

Figure 3: Example of a well-formed and not a well-formed XML document

Typical XML document consists of two parts: processing instructions and body of the document. The first line is the processing instruction for an XML parser, which specifies that this document applies XML version 1.0 and 8-bit Unicode encoding. [6] The second part is a body of the document which consisting of the root element – outermost element of the documents structure and child elements containing the data of an XML file. Between tags of is situated the content of an element. There are no restrictions for the naming of an element, except the rule, that white spaces are forbidden. [7] As it is shown in the example, in a well-formed document should be only one root element, each element should have closing tag or it can be self-closing if it is empty as it is shown on the Element3 of the example and should be properly nested (wrong version - <Element1><Element2>This document is not well-formed</Element1></Element2>). Elements can include attributes,

which containing metadata as shown in the example. [5] In a well-formed document two attributes must have different values, for example, if a document contains such attributes `<Element2 Attribute1="1.0" Attribute1="Data">` parser will end up with a fatal error. [8] The physical structure of an XML document is concentrating on the entities, and it is not covered in this thesis.

These rules are creating the key advantages of the language. XML is the standardized industrial language which is vendor-independent, which made it popular in the industry. It is self-describing and easily can be read by humans and machines. The most important benefit of the language is extensiveness, in other words, the possibility of creating own markup tag for structuring a data. The combination of the previous benefits creating the new benefit: efficiency of the processing and transfer of the data. This benefits made XML popular in the software development industry, and a high number of the software are able to read the documents to some degree. [6]

Large popularity of the language made it applicable to different fields of applications, such as military, business and communication protocols fields. [7] In the multibody simulations it is used as an application independent data asset for storing and transfer of the simulation parameters and results for the further development. It is used in the Mevea real-time simulation software, which is used partially used in this thesis.

2.3.1 Structure of the simulation input XML document

Mevea is real-time simulation software developed by Mevea Simulation Solutions Ltd., which headquarters is located in Lappeenranta, Finland. This software package is specializing in the multibody dynamics and hydraulics real-time simulation. Software package consists of four programs, but this thesis used only Solver and Modeler. The solver is software, which loads and running a simulation. The modeler is software, which is used for development of the model, environment and other features of a simulation. [9]

During the modeling stage in the Mevea Modeler software are generated XML files, which are used for the input of the initial parameters for the Solver. There is the main simulation file with mvs extension, which contains model and world files. Structure of mvs file shown in figure 4.

An MVS file is the file format, which is based on the XML and used as a starting directive for the Solver software. It is automatically generated after saving a model in the Modeller software and contain data about the environment, world and model.

```

<MeVEASimulation>

  <Environment simulationName="New Simulation" nVisualizations="1"
  useParticles="No" particlesInThread="Yes" particleCellsInThread="Yes"
  collisionGroupDefinitionsFile="" nCollisionThreads="4" />

  <Models ModelFile="Example.xml" solverID="1" useArchive="No"
  MPKFile="" hydraulicsSchematicFile="" />

  <World WorldFile="Example_world.xml" />

</MeVEASimulation>

```

Figure 4: Structure of Mevea MVS file

Environment element contains attributes, which are specifying the main parameters related to the environment. World element consisting of the reference to the world XML file. The model file is creating high interest for this thesis topic because it contains data about the structure of the model and its parameters.

The model file consists of a root element called DMInputFile and seventeen child elements. Each of the elements in the file is representing the model feature. After addition of the model feature in the Mevea Modeler program generates a new child element of the type of asset, for example, if the body was added, it would create child element of the bodies element. The elements of the file are: Model, MeVPacker, ExternalInterface, Splines, Bodies, Dummies, Constraints, Forces, Electrics, Graphics, Collisions, DataSources, ObjectMovers, Inputs,

Outputs, MotionPlatformControl and SoundComponents. In this thesis will be discussed commonly used elements, which are model, bodies and forces.

The model element consists of one child element and twelve attributes. Structure of the element shown in figure 5.

```
<Model Name="Wheel_loader" Method="4" Integrator="1"
SparseSolver="No" dt="0.0015" tstop="3e4"
update_frequency="30" error="0.001" hydrIntMethod="0"
numHydrIntegr="10" nGSIterations="3" logoImageName="">
  <Gravitation x="0" y="-9.81" z="0" />
</Model>
```

Figure 5: Model element structure of DMInputFile

It contains data about the vector of the gravity in the gravitation element. The attributes of the element are describing the parameters of the simulation, such as the name of the simulation, solution method, numeric solver type, using of the sparse solver, time step, overall simulation time, hydraulic integrator type, hydraulic integration steps, the maximum number of iterations and logo of the file.

Bodies element consists of child elements, which representing each body in the simulation. A typical representation of the element is shown in figure 6.

```

<Bodies>
  <Ground UsePF="No" RelTo="Ground" BodyType="Rigid"
  VisualizationGraphics="Graphics_Ground;Graphics_Sky;Graphics_Envir
  onment" CollisionGraphics="">
    <Position x="0" y="0" z="0" />
    <Orientation phi="0" theta="0" psi="0" />
    <Inertia mass="0">
      </Inertia>
      <I Ixx="0" Iyy="0" Izz="0" Ixy="0" Izx="0" />
      <CM x="0" y="0" z="0" />
      <CMI x="0" y="0" z="0" />
    </Ground>
  </Bodies>

```

Figure 6: Structure of Bodies element in DMInputFile

On figure 4 shown the simplified example of the element, containing only one body, which representing ground. It consists of three elements and five attributes. Elements are describing position, orientation and inertia properties of the body, such as moments of inertia, the center of mass and inertia frame definition.

The forces element consisting of five elements and one self-closing element, representing the hydraulic bulk modulus. The element structure is shown in figure 7.


```
<Forces>  
  <Tyre> </Tyre>  
  <B2BM> </B2BM>  
  <B2BF> </B2BF>  
  <Motor> </Motor>  
  <PowerTrain> </PowerTrain>  
  <Hydraulics Oil_B0="1600000000" />  
</Forces>
```

Figure 7: Structure of Forces element in DMInputFile

Each element representing different types of forces, which are tyre models, body-to-body torque, body-to-body force, motor and powertrain (gearboxes, differentials, planet gears). The tyre model is responsible for creating tyre contacts, which are taking place between tyre and other bodies in the space. The body-to-body force and torque are representing forces and torques, which are acting between two bodies in the space. The motor element defines parameters of the motor such as torque, idle speed and state of the motor. The powertrain element defines initial parameters of gearboxes, differentials and planet gears of a machine. As it can be observed, these files are including initial data of the simulation, which is static and is not changing during the simulation runtime. This is the main factor, which will be used in this thesis for the controlling parameters of the machine.

2.3.2 Parsing and connection to the interface

The XML file should be analyzed with special software called parser. The parser analyzes the structure of an XML file and extract needed variables. In this thesis will be parsed.mvs file and model XML files. For the parsing targets in the model XML file were chosen model, bodies and forces elements, because they have the highest effect on the model behavior. The simulation parameters are essential to change, for increasing of the responsiveness, adaptivity and creating a possibility for future debugging. The bodies and forces parameters are the core of the user interface because it will provide the flexibility of the simulation and create a possibility to modify the mathematical model by a user visually, for example, by increasing of the body size. The parameters, which could be changed are: integrator parameter, time step, update frequency, bodies mass/inertia properties, motor and transmission parameters.

2.4 Requirements for User interface

The user interface of the real-time simulation is a complex problem, which needs to consider multiple aspects of the program. In this chapter will be discussed requirements for the operation of the simulation, design requirements and aspects related to the comfortable using of the application with head mounted display. During the work, the author defined following aspects: user communication, reading/writing data from XML files, saving user-specified data for the subsequent analysis, protecting the simulation device from abuse, starting Mevea Solver process/launching/pausing/stopping the simulation.

2.4.1 Communicating with the user through crafting menu

The main functionality of the user interface is gathering, storing and transfer of the data from a user in a simple and ergonomic form and displaying simulation parameters. For controlling of the simulation were created two user interfaces: the main menu and the in-game menu. The main menu is responsible for data inputs for XML files. The machine model from model XML files must be taken through a series of sliders and buttons. Users will have the possibility to create their version of a machine with unique parameters of transmission, engine and hydraulic forces. In the main menu, it is possible to use presets of machine parameters which will be provided by a separate set of XML files configured by the author. In these presets will be different parameters of meshes and bucket sizes, which can't be modified by XML files. Corresponding mass and inertia parameters of bodies will be

specified in XML files related to the meshes preset. In the main menu will be asked demographical information about the user, which will be explained in the next chapters. The in-game menu will give access to pausing, stopping, restarting the simulation, and returning to the main menu. The controls of the menus must be intuitive and comfortable to the user, and they need consideration of multiple design aspects.

2.4.2 Comfort level of the view

The design of the user interface is constrained by human physiology and technical requirements of the used hardware. The first aspect, which needs to be considered is a comfortable position of the head and body of a user. At figure 8 shown comfortable and maximum angles of the user's head rotation angles. The comfort zone in the horizontal plane is 30° from the head center, in the vertical direction this angle is equal to 20° upwards and 12° downwards from the center [10]. At the maximum angles of rotation, the user starts feeling slight discomfort [10]. On the right side of figure 8 shown an angle, when user intent to rotate the body to see objects located at the marked area. In summary, the most comfortable area to place objects inside the scene is in 60° zone in the horizontal plane and 32° in the vertical plane.

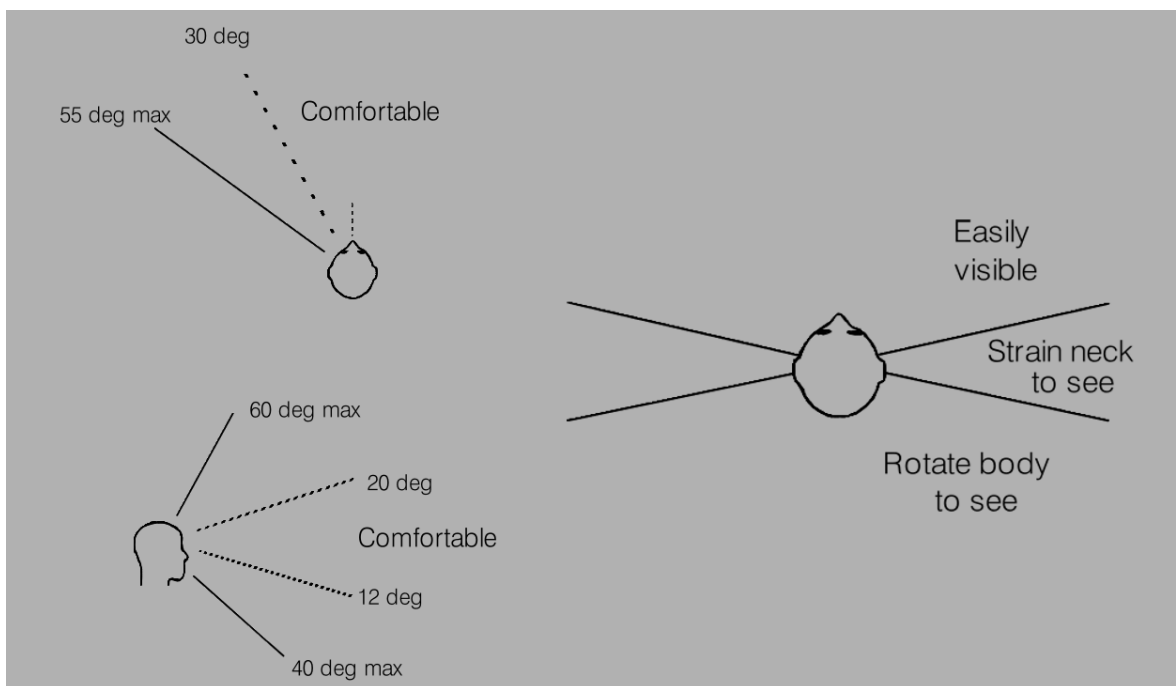


Figure 8: Comfortable and maximum areas of view [10].

Another important matter related to the field of view of hardware is the perception of depth, which shows human eye focus distance. The graph of the perception of depth for Oculus Rift is shown on figure 9. From the figure can be observed that at a distance lower than 0.5 meters starts eye strain and cross-eye zone [10]. After 20 meters 3D objects are losing their depth and look like flat objects. These zones must be avoided while designing the user interface. The desirable area for placing the user interface inside virtual space is the area between 1 to 10 meters. For the further design must be considered application of Leap Motion controller, which is tracking hands movement and create arms meshes inside the virtual reality. For the comfortable using of the Leap Motion controller, distance to the user interface's intractable objects must be no longer than the length of the arm. It is approximately 60-65 centimeters. According to the previously listed information, the user interface must be divided into two parts: controls such as buttons and sliders must be closer than 60 centimeters and meshes/text must be placed in the area between 1 to 10 meters. These requirements must be applied to the real-time visualization during the simulation.

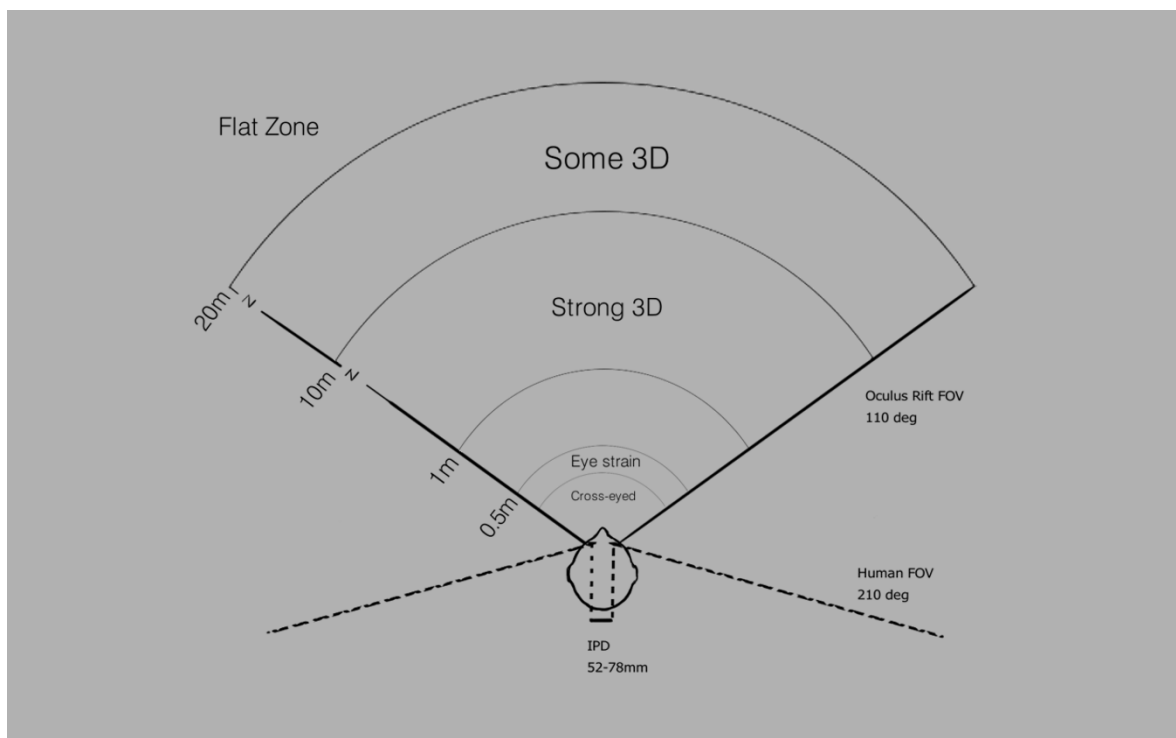


Figure 9: Perception of depth for Oculus Rift HMD [10].

2.4.3 Visualization

Visualization functionality is added to the application for convenience of using and due to Mevea software limited visualization capabilities. Mevea Solver is not supporting Oculus Rift virtual reality headset, advanced lighting techniques and post-processing functionalities, which is applied in this thesis. Without virtual reality visualization of the simulation, the user must take off the headset and continue to work with light emitting diode (LED) display, what is unreasonable. The visualization is performed by transferring model's bodies positions to the application via User Datagram Protocol (UDP) IP socket. These positions will be assigned to the actors inside the application, which will create a visualization of the bodies. For each body representing graphics object will be created graphical materials assets and custom shaders. For the visualization of the environment will be applied graphical materials as well, with subsequent post-processing such as aliasing, lens flare, precipitation and particles. These features will create the realistic view of the simulation and provide full immersion with subsequent improvement of the interactivity.

2.4.4 Interaction

The game engine is providing wide opportunities for development of reach interaction between user and application. By applying of the game engine is possible to create a highly interactive main menu/in-game menu, a machine cabin and tasks for an operator.

For the simulation can be created complex tasks with a higher level of the user performance control. During the tasks, a user can get guidelines, advice and achievements. In this thesis will be used the task of loading a truck with construction debris. The user should clean a parking slot and yard area without hitting objects around, such as cars, houses and other environment parts. Debris must be placed in the specific area in the yard. After finishing the first task, the user must take debris from the created pile and load a truck. User performance will be evaluated by the scale from zero to five, which will be calculated from the cleaned area, amount of debris loaded in a truck and number of hits. There will be a possibility to skip one of the tasks at the main menu with the addition of penalty points.

The main menu can be improved by animations and customizing tasks, such as assembling parts of the machine, guidelines for driving and maintenance in the form of the document

and video. The machine cabin can add immersion effect by applying tasks for of the cabin parts such as buttons, levers, radio and windows with an actual response (turning on blinkers, headlights, radio/switching station and other tasks). The in-game menu can improve interactivity by the modifying some parameters during the simulation, task selection possibility, environment changing and other features. This kind of interaction provides a compromise between the responsiveness of the simulation to the user actions and controlling his/her behavior.

2.4.5 Security requirements

The simulation studio is consisting of two Mevea motion platforms. These motion platforms are consisting of personal computer hidden inside of the platform, an adjustable chair, control hardware, LCD and motion platform with hydraulic actuators. Motion studio hardware can be observed in figure 10. Control hardware includes two joysticks, buttons simulating some functionalities of a keyboard, steering and pedals. The platform is equipped with Mevea Launcher software, which is working during the operation of the platform and turned on automatically by a particular process.

The security requirements are created based on the previous experience and problems, which were faced during the operation of the simulation studio in Lappeenranta University of Technology. Due to the software and hardware configuration were faced multiple problems with security, malfunctions and abuse. Commonly, users of the platform were able to close or hide the Mevea Launcher with tab, directions and enter buttons. For that reason, were created a process, which is described previously, but it was stopped by some users as well. Commonly, users are opening a browser and accessing websites. This is the dangerous situation because the user has access to the system and programs inside the machine.



Figure 10: Structure of motion platform: 1-LED display, 2-steering wheel, 3-joysticks, 4-pedals, 5-control buttons, 6-seat.

One of the significant risks, which must be fixed in the user interface is lack of the security and restricting the user from controls of the simulation studio. That is the reason to refuse the application of the installed input devices on the simulation studio and use alternative inputs, which are connected to the user interface application. Instead of the default buttons must be used for ergonomic and straightforward devices. For the current version of the user interface is used a combination of Leap Motion, default steering wheel and joysticks. This input setup will provide a controlled environment, which will not give the possibility to access excessive features of the system. At the same time, this kind of setup creating a problem with loading simulation, because application of Mevea Loader was rejected. There will be no possibility to load the simulation independently by the user.

2.4.6 Loading and running the simulation

As it was discussed previously, the user interface has two main functionalities. They are communicating with the user through a crafting interface and visualization of the real-time simulation. At first, the user is asked to configure the machine for the real-time simulation inside VR and then the simulation should be visualized with Unreal Engine. For the visualization of the simulation were used sockets, which requires the following conditions:

1. Mevea and Unreal Engine 4 are running at the same time
2. Server and Client have a connection
3. The simulation should be started.

These conditions create a problem with starting of simulation. A user cannot start simulation manually because the user does not have access to computer inputs. For solving of the problem, the application should be able to load the file, start, stop, reset, pause the simulation and close simulation. At the end of the crafting user will push the button “Start simulation”, which will save all changes and start the simulation. On the next step, the application is creating Mevea process in the system. Then, it will open the .mvs simulation file. For controlling the simulation will be used tasks features of Mevea software, which should be controlled by Python script.

The application must recognize the situation when Mevea Solver is not able to open the file and notify technical staff about malfunction via e-mail. The user activity must be tracked for preventing using the same session by multiple users. This can happen when the next user is coming to the simulator and the previous user forget or intentionally didn't stopped the simulation. The application should stop the simulation after a specific time, for example, two minutes without the activity of the Oculus sensor and other inputs such as joysticks. When the simulation is stopped, all the data about the session will be saved.

2.4.7 Saving user-specified data

Saving and storing user-specified data for its further analysis is one of the essential functionalities of an application. The user interface must collect anonymous data for a business and development departments about modification and performance of the model during simulation. The data will be collected from the main menu, where the user will specify demographic data such as sex, age and occupation. The data about the configuration of the model from the main menu, such as the parameters of the model and simulation, will be collected and saved automatically. The data about model performance will be transferred from the Mevea Solver software via UDP IP socket and stored at the other file. Also, it is possible to define users, who were using the simulator before by tracking if the tutorial was skipped.

2.4.8 Providing tutorials and additional information

It is essential to provide a tutorial for users, which are using the application for the first time. The author assumes, that most of the target audience of the machine is not familiar with the concept of the virtual reality and using of the related devices, such as Leap Motion. The tutorials must be divided into two parts, which are explaining how to start the simulation and how to operate the simulation inside the virtual reality. All the tutorials must support two languages, Finnish and English. Tutorials must explain the control and meaning of some actions or parameters in the straightforward language. The most preferred type of the tutorials is a visual representation of the needed actions to perform.

The first part, further in the text preliminary, will explain to the user how to start the simulation, check an environment/machine before the simulation start and advise about comfortable using of the machine. It must be made in the form of the video, which will be played on the LCD screen near the loading station. In this video will be shown a person, which sets up the environment for using text instructions below. At first, should be presented the information about setting up the motion platform: how to set up chair height, distance to the wheel and precaution about save zone area around the machine. After must be explained the procedure of the Oculus Rift wearing and setting up of the lenses distance. At the end of the video, the user will be guided to wear the VR headset and follow the procedures in the next part of the tutorial.

In the second part of the tutorial will be explained controls of the user interface and in-game controls. For the user will be presented Leap Motion controller and how it is interacting with the application. After explaining the operation of the Leap Motion controller, the user will get guidance about options, which can be changed in the machine parameters and how they are affecting the machine behavior.

Also, the user will be provided with optional tutorials about machine structure. It will include a description of machine parts, such as the configuration of transmission, engine and additional parts, for example, buckets. For the user will be provided a 3D model of the internal machine structure and explained the basic operation of each part. Optionally, the user can familiarize with documentation and promotional materials in the specific sections of the interface.

2.4.9 Control of other parameters: graphics, controls and appearance

To the interface will be added the possibility to adjust other parameters, which are related to the graphics, controls and appearance. In this section, the user can define the type of the output: Oculus Rift or LED display. If the user chooses LCD output, the application will request to connect a mobile device for input of starting parameters and control of the simulation during runtime. If the display option is chosen, the user will have the possibility to adjust graphics parameters, such as quality, post-processing parameters and resolution.

2.4.10 Hardware, software and system requirements

For the most comfortable experience frame rate must not be least than 60 frames per second. This requirement can be met by using the right components on a used computer. The computer, which will run the application should meet the recommended specification to used hardware and software. The application uses the Leap Motion controller and Oculus Rift headset.

Oculus Rift headset minimum and recommended specifications are listed at table 1. This hardware requires middle range gaming graphics adapters, which have more than 1.3 GHz clock speed and more than 4GB memory size. The processor should be able to load the graphics adapter for the best performance of the graphics card, that is why should be used processors with a clock speed higher than 3.4 GHz, more than two cores, four threads and last level cache size more than 3MB. In summary, the graphics adapter is playing the leading role in the Oculus Rift performance.

Table 1. *Oculus Rift hardware and OS requirements.*[11]

Hardware	Recommended Spec	Minimum Spec
Graphics Processing Unit (GPU)	NVIDIA GTX 1060 / AMD Radeon RX 480 or greater	NVIDIA GTX 1050Ti / AMD Radeon RX 470 or greater
Alternative graphics card	NVIDIA GTX 970 / AMD Radeon R9 290 or greater	NVIDIA GTX 960 / AMD Radeon R9 290 or greater

Table 1 continues. *Oculus Rift hardware and OS requirements.* [11]

Hardware	Recommended Spec	Minimum Spec
Central Processing Unit (CPU)	Intel i5-4590 / AMD Ryzen 5 1500X or greater	Intel i3-6100 / AMD Ryzen 3 1200, FX4350 or greater
Random Access Memory (RAM)	8GB+	8GB+
Video output	Compatible HDMI 1.3 video output	Compatible HDMI 1.3 video output
USB ports	3x USB 3.0 ports plus 1x USB 2.0 port	1x USB 3.0 port, plus 2x USB 2.0 ports
OS	Windows 8.1 or newer	Windows 7 SP1 64 bit or newer

Recommended requirements for Leap Motion controller are shown at table 2. From the table, we can observe that the most critical parameters are related to the processor, random access memory and type of USB port.

Table 2. *Leap Motion hardware requirements.* [12]

	Recommended Spec
CPU	AMD Phenom™ II or Intel® Core™ i3
Hardware	USB 2.0 port
RAM	2 GB
OS	Windows® 7/8 or Mac® OS X 10.7

Hardware and operation system requirements for Unreal Engine are specified at table 3. These requirements are applied to the Unreal Editor 4.18.3. As long as the application will be packaged and build as an independent executable file, it needs further benchmarking, because graphics assets and interactions can load the system higher.

Table 3. *Hardware and OS requirements for Unreal Engine 4.*[13]

	Recommended Spec
CPU	Windows 7/8 64-bit
GPU	DirectX 11 compatible graphics card
RAM	8 GB
OS	Windows 7 SP1 64 bit or newer

According to the previous information, can be created a preliminary minimum and the recommended specification for the simulation computer. It can be observed at table 4.

Table 4: *System requirements for user interface.*

Hardware	Recommended Spec	Minimum Spec
GPU	NVIDIA GTX 1060 / AMD Radeon RX 480 or greater	NVIDIA GTX 1050Ti / AMD Radeon RX 470 or greater
CPU	Intel i5-4590 / AMD Ryzen 5 1500X or greater	Intel i3-6100 / AMD Ryzen 3 1200, FX4350 or greater
RAM	8GB+	8GB+
Video output	Compatible HDMI 1.3 video output	Compatible HDMI 1.3 video output
USB ports	3 USB 3.0 ports and 2 USB 2.0 port	1 USB 3.0 port and 3 USB 2.0 ports

Summary of the requirements to the user interface is shown at table 5.

Table 5: *The overview of the user interface requirements.*

Functional requirements	
Communication with user	<ul style="list-style-type: none"> • Must collect user-specified parameters of the machine via parametrization menu
Comfort level of view	<ul style="list-style-type: none"> • The interface must be located not more than 60° from normal in the vertical plane

Table 5 continues. *The overview of the user interface requirements.*

Functional requirements	
Comfort level of view	<ul style="list-style-type: none"> • The interface must be located not more than 55° from normal in the horizontal plane • All valuable meshes related to controls must be located in the range from 1 to 0.5 m from the user location
Visualization	<ul style="list-style-type: none"> • Must provide visualization of the machine movement • Must provide machine status during the simulation • Must provide visualization of environment
Interaction	<ul style="list-style-type: none"> • Must contain task inside of the simulation related to moving objects/substances
Security requirements	<ul style="list-style-type: none"> • Must restrict access to the system
Loading and running the simulation	<ul style="list-style-type: none"> • Automatic simulation start/pause from the interface • Automatic closing of Mevea Solver from the interface
Saving user-specified data	<ul style="list-style-type: none"> • Obtain and save demographical data • Obtain and save machine parameters • Obtain and save applications log data (error handling)
Type of user interface	<ul style="list-style-type: none"> • Graphical interface with Leap Motion input
Hardware, software and system requirements	<ul style="list-style-type: none"> • Software: Oculus application, DirectX 11, Windows 8 OS

Table 5 continues. *The overview of the user interface requirements.*

	Hardware: Intel i3-6100 or greater, NVIDIA GTX 1050Ti or greater, 8GB RAM, 1 HDMI 1.3 port, 4 USB 3.0 ports
Non-functional requirements	
Providing tutorials and information	<ul style="list-style-type: none"> • Video tutorial on starting the simulation and structure of hardware used • In-game tutorial about using the interface
Communication with user	<ul style="list-style-type: none"> • Should contain two menus: main and in-game • The menu must consist of sliders and buttons • The in-game menu must contain controls of the simulation (start, stop, restart, return the main menu)
Visualization	<ul style="list-style-type: none"> • Realistic environment • Realistic meshes for representation of machine bodies • Apply post-processing effects
Security requirements	<ul style="list-style-type: none"> • Restrict user controls • Control user behavior and prevent abusing or dangerous behavior
Control of the other parameters	<ul style="list-style-type: none"> • Add control of the output type: LCD or Oculus Rift

These specifications are mostly affecting graphics performance of the simulation and user interface application. Nevertheless, this hardware can affect the overall performance of the simulation, because of the simulation working principle, which is discussed in the next chapter.

3 RESULTS

3.1 Operation of the user interface

The main purpose of the interface is gathering of the user preferences for subsequent application of them in the development process of a product. For a correct gathering of the data, a customer must finish the simulation from the start to the end. If the data collection process is interrupted, it is corrupting the sample and makes it not applicable to the development of the new product. The effect of the simulation interruption is shown in figure 11. The interruption of simulation after different stages of the simulation may result in data loss. If the configuration menu stage was completed but stopped in the middle of the simulation stage, machine behavior and feedback data will be lost. If the simulation was completed but stopped before entering feedback, customer feedback will be lost.

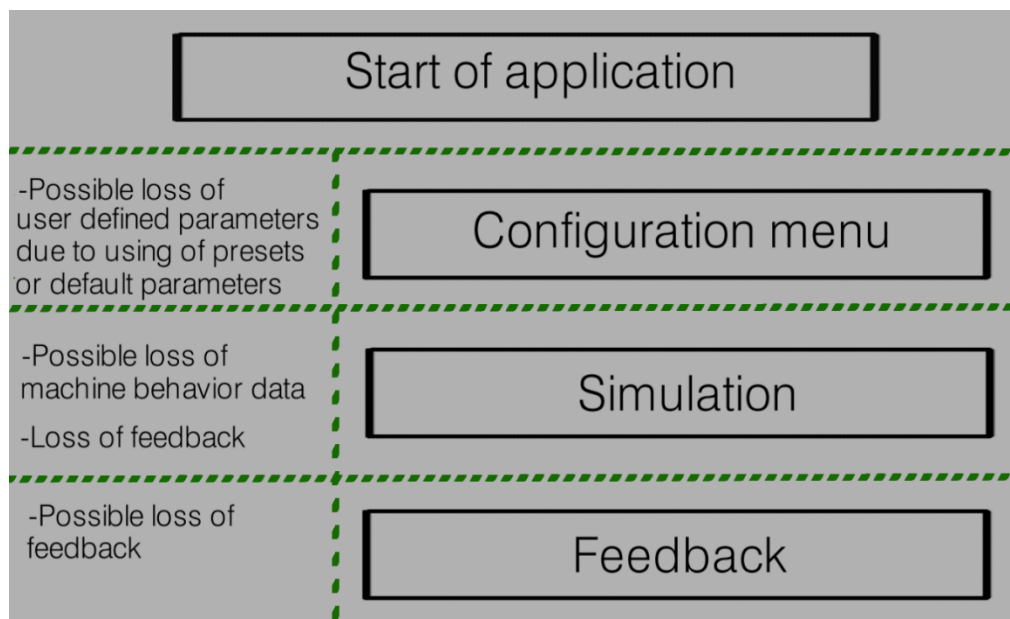


Figure 11: Data loss due to simulation interruption

3.1.1 Application performance

The user can interrupt the simulation by the multiple reasons, but the most probable reason is virtual reality sickness due to the low performance of the simulation. By the low performance of the simulation is implied visualization performance, which can lead to the immersion disruption and virtual reality thickness. The user interface, described in this thesis

work, is a complicated application, which includes multiple processes and components. The majority of the processes inside of the application require high update rate. The most crucial part, which requires high performance is real-time simulation and visualization of the machine. When the machine is simulated, two programs are involved: Mevea and Unreal Engine. Data transfer between these two applications is bound to the so-called tick function, which is called at each frame change and strongly affected by the graphical performance of a machine, graphics complexity and delta time of the application. Mevea Solver and UE4(Unreal Engine 4) have a different load to the GPU of a computer because they have the different architecture of the visualization, for example, shaders and post-processing effects. For meeting requirements for most comfortable using, their work must be synchronized regarding frame rate. The issue is solved by the application of the computer with higher specifications. This computer has Intel i7 processor, HyperX 16 GB RAM, Nvidia GTX1080 GPU and located at Lappeenranta University of Technology laboratory.

The same matters are applied to the main menu part of the program. As long as the main menu is made in virtual reality, it must be able to work with a constant frame rate not less than 60 frames per second. The main menu of the application is an important part of the simulation because at this stage obtained more than a half of the information, namely user demographical data and user-specified parameters of the machine. Nevertheless, during this stage considerably small XML file is parsed, that is why most of the problems were created due to graphics complexity. The load to the system is less compared to the simulation real-time simulation process because modification of an XML file does not require work of Mevea Solver at the same time with the UE4 application. Problems with graphical performance were fixed in the same manner as in simulation part, by application of more powerful computer and graphics assets optimization. The other matter from which graphical performance depends is the overall architecture of the application.

3.1.2 Data input method

At the start of the program user required to input data to the simulation. It is performed by application of Leap Motion controller shown at figure 12. The sensor tracks position and rotation of the user's hands for interacting with an application. It is connected to the head mounted display. This layout makes an illusion of presence inside a virtual scene.

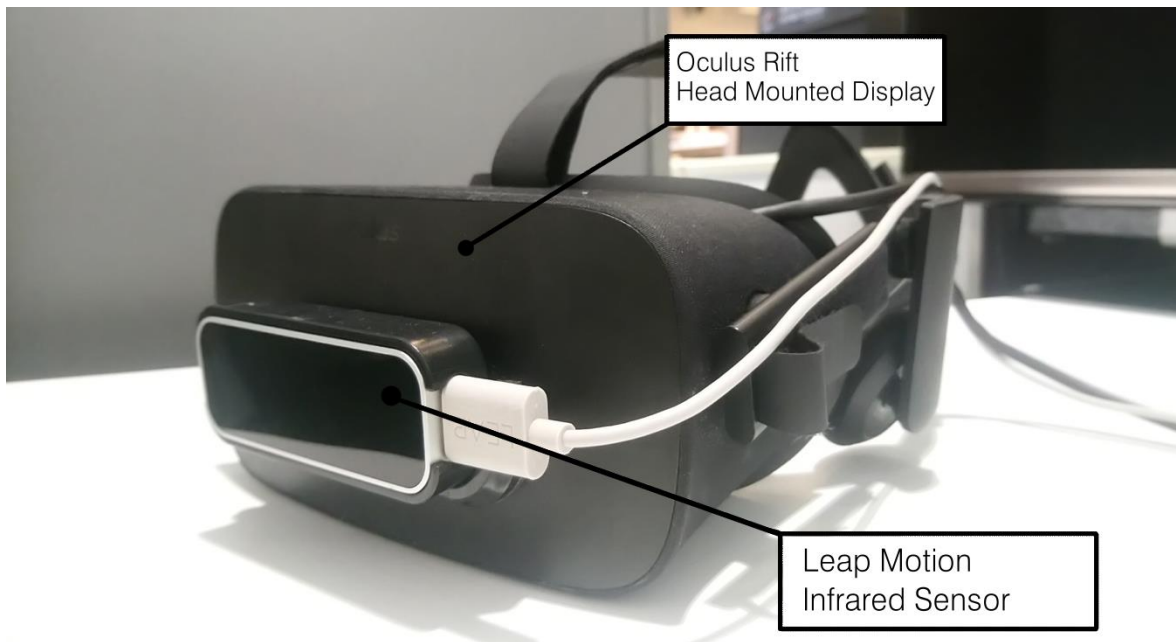


Figure 12: Leap Motion controller on the head mounted display

The specially designed blueprint tracks the position of the actor and physics-based mesh for further calculation of the parameter value. Mesh is constrained along the y -axis and z -axis. The application is calculating parameter value by the following formula:

$$VAL_{p1} = |X_a - X_m| \quad (1)$$

This layout of the user input has minor drawbacks. The first drawback is related to the operation of the interface. It is possible, that after placing the actor at the origin, an incorrect value is obtained. This situation happens, because in this layout the initial position of the mesh is in the minus zone, but the final value may be in the positive zone. This situation was solved by simply placing the actor aside from the origin. Another drawback is related to the operation of the physics inside the game engine. If the box mesh, shown on figure 13, pushed from the side, it behaves correctly. If user tries to grab the mesh, it starts to wobble. This drawback was fixed by applying event-driven code, which is activated after grabbing of the mesh. The final prototype version is shown in figure 13.



Figure 13: Prototype of the parameter changing menu

3.1.3 Main menu operation

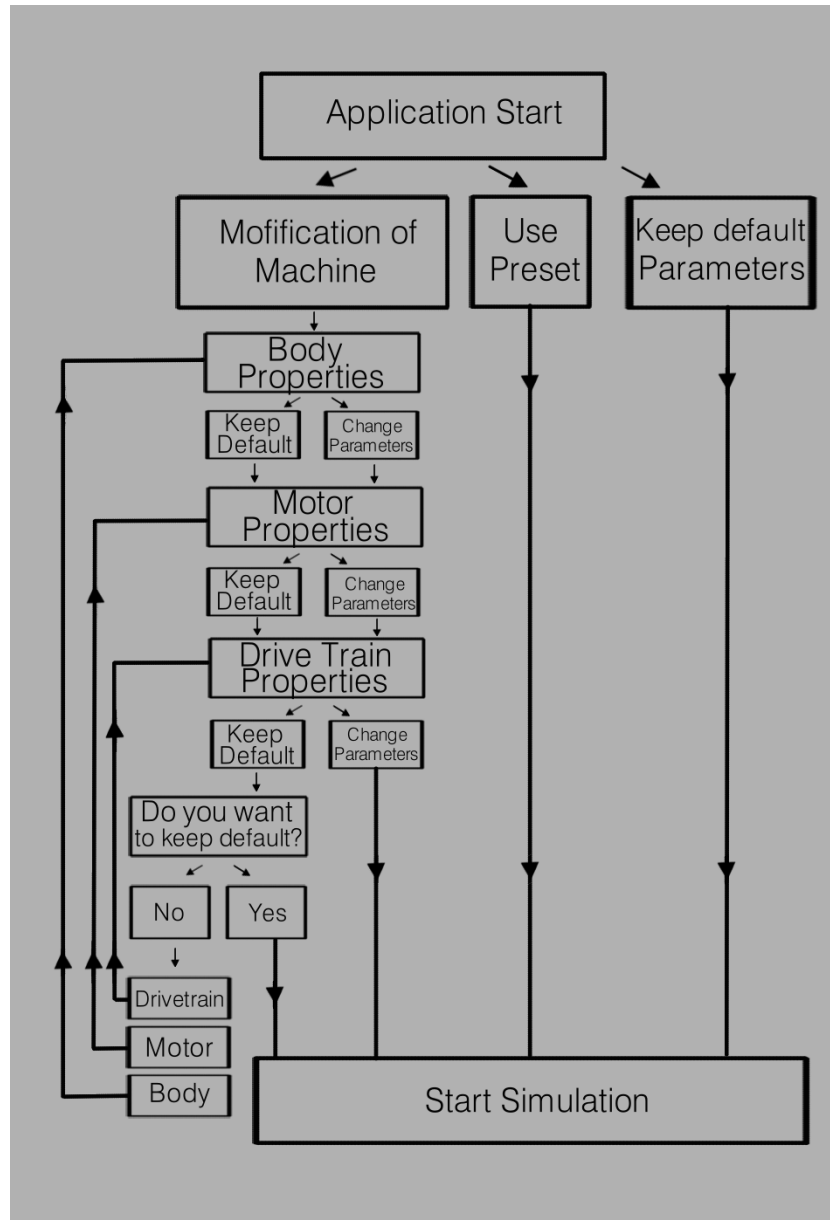


Figure 14: Behavior tree of the main menu

At the start of the application for the user has three options for starting of the simulation. In figure 14 can be observed a tree diagram, which shows possible interactions with the menu. In general, there are three options, namely, change parameters, use presets and keeps default parameters. At the modification root, user will be asked to change the parameters of the model or leave them as default. The user can change the parameters of bodies, motor and drivetrain parameters of the machine. The user has an option to change parameters, which

are shown at table 5. All parameters have own range, which restricts the user from choosing insufficiently low or high value causing improper model behavior.

Table 6. *User defined parameters options.*

	Parameter	Range	Units
Motor			
1	Torque curve (each point)	100-750	Nm
2	Idle speed	60-80	rad/s
Transmission			
3	Gear ratio, gear 1	3.85-4.15	-
4	Gear ratio, gear 2	1.9-2.20	-
5	Gear ratio, gear 3	0.95-1.25	-
6	Gear ratio, gear 4	0.45-0.75	-
Mass properties			
9	Front body mass	1800-2400	kg
10	Rear body mass	5000-7000	kg
11	Tyres mass	250-350	kg

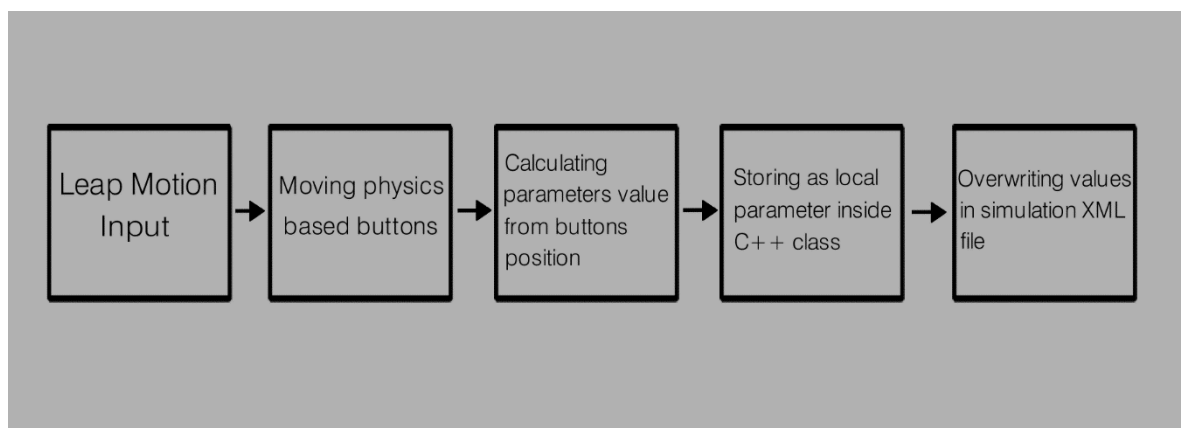


Figure 15: Changing parameters menu operation

After defining these parameters, they are assigned to the corresponding attribute in the XML file, which is containing the model parameters. Assigning of parameters is made by special C++ class, which is based on the TinyXML parser. Working principle of the class is shown in figure 15. Unreal Engine application is receiving input from Leap Motion controller and assigning user's hands position to a pawn. The event-driven code defines the moment, when

the pawn is touching a button on the user interface and defining parameter's value. The next part of the code assigns obtained variables to local parameters inside the class, open the XML file and overwrite it with new parameters. Then, when the simulation file is opened, new values of the model parameters are applied. If the user, during modification root, choosing to leave all parameters as default, the application will ask if this is a right input. If the user specifies, that he/she wants to change the parameters of the machine, the application is asking which part should be modified.

In certain cases, it is possible, that user is not intended to change model parameters, but not he/she is not satisfied with default settings. Then, the application gives an option of the parameters presets. Figure 16 showing the procedure of the presets handling. The user is specifying in the menu, which kind of parameters preset should be used. According to the user's choice, the application finds appropriate XML parameters file and set it as main in the .mvs file.

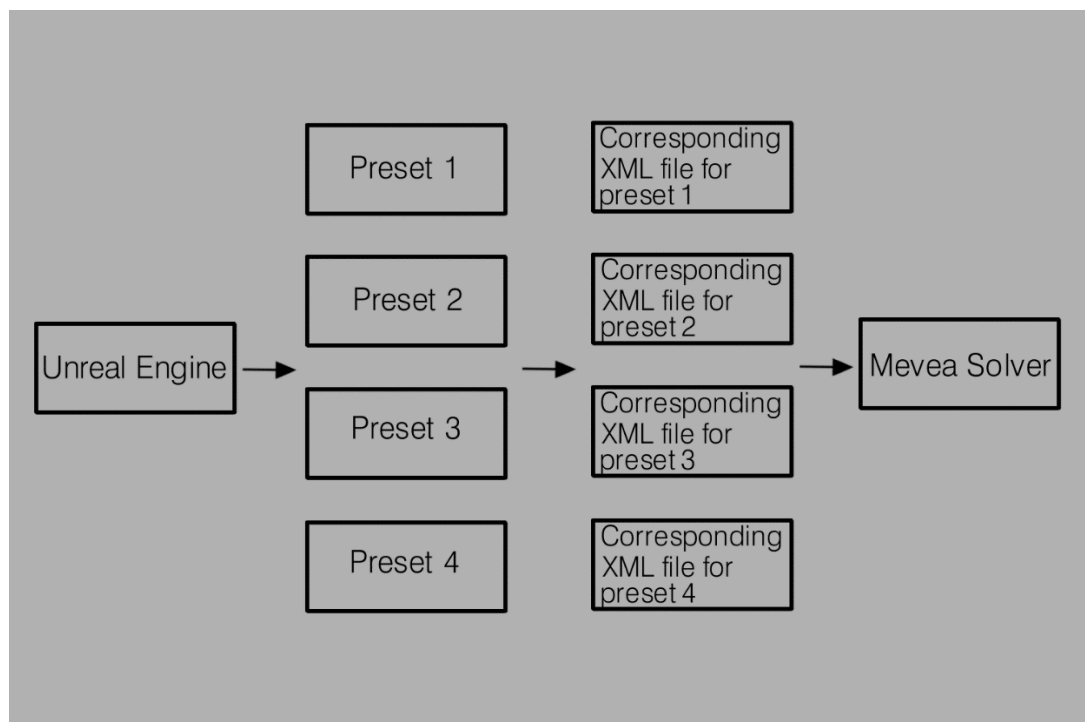


Figure 16: Presets choosing behavior tree

After filling up form of the machine parameters or choosing a preset, application asks user to input demographical data, which will be explained later. When the initial data is obtained, the simulation process starts.

3.1.4 Simulation operation

For a better understanding of the simulation operation, it is important to define the topology of the machine model. The model consists of 9 bodies, 11 joints, has 54 generalized coordinates, 50 constraint equations and 4 degrees of freedom. The model structure is shown in figure 17. As long as Mevea model is responsible for all physics and collision, it must contain all bodies and dummies for the correct representation of the real machine. At the same time, the other reason for a detailed representation is a possibility of changing machine parts' parameters, which can be used in the subsequent studies.

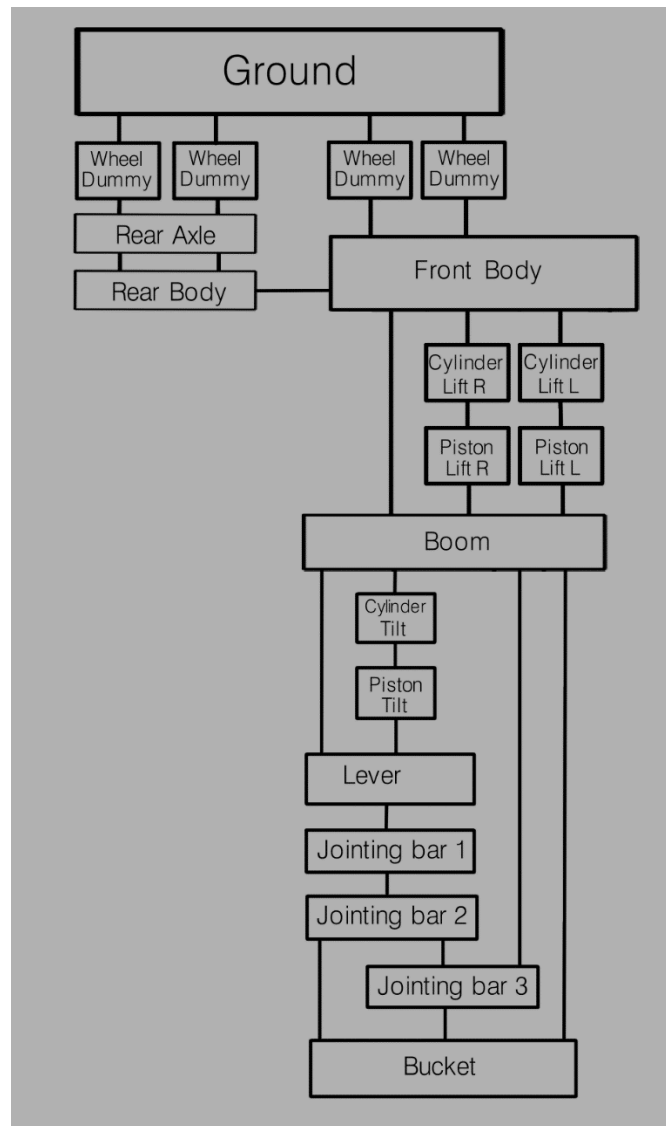


Figure 17: Mevea model topology

The model in Unreal Engine 4 is made only for the visualization purposes, that is why it should be simplified. The model will contain only four bodies, which will be used for visualization of the model movement. They are a front body, a rear body, an axle and a bucket. The rest of the bodies and dummies (meshes, which represents actuators and force components) are connected to these bodies. Wheel dummies are connected to the front and rear bodies, from which they are taking rotation and position in the world. For creating of a wheel, the rotation was used rotational velocity of each wheel. Parts, which are connected to the bucket and front body was made by rigged animation. The other reason for the model simplification is effective data transfer between two application for excluding possible troubles during the simulation.

The first thing, which is performed by the application is starting of the Mevea simulation file. The simulation file is opened while a user is transferred from the main menu map to the simulation map. When a user clicks the start button, simple code initiates the Mevea Solver process and opens the needed simulation file by its relative path. After opening the file, the data transfer begins.

During the simulation, Mevea solver sending the information about the machine to the UE4 for setting inputs, actor position/rotation or displaying the current state of the machine. Mevea and Unreal Engine are exchanging the information in order to create a visualization of the simulation. Figure 18 represents the data exchange between applications.

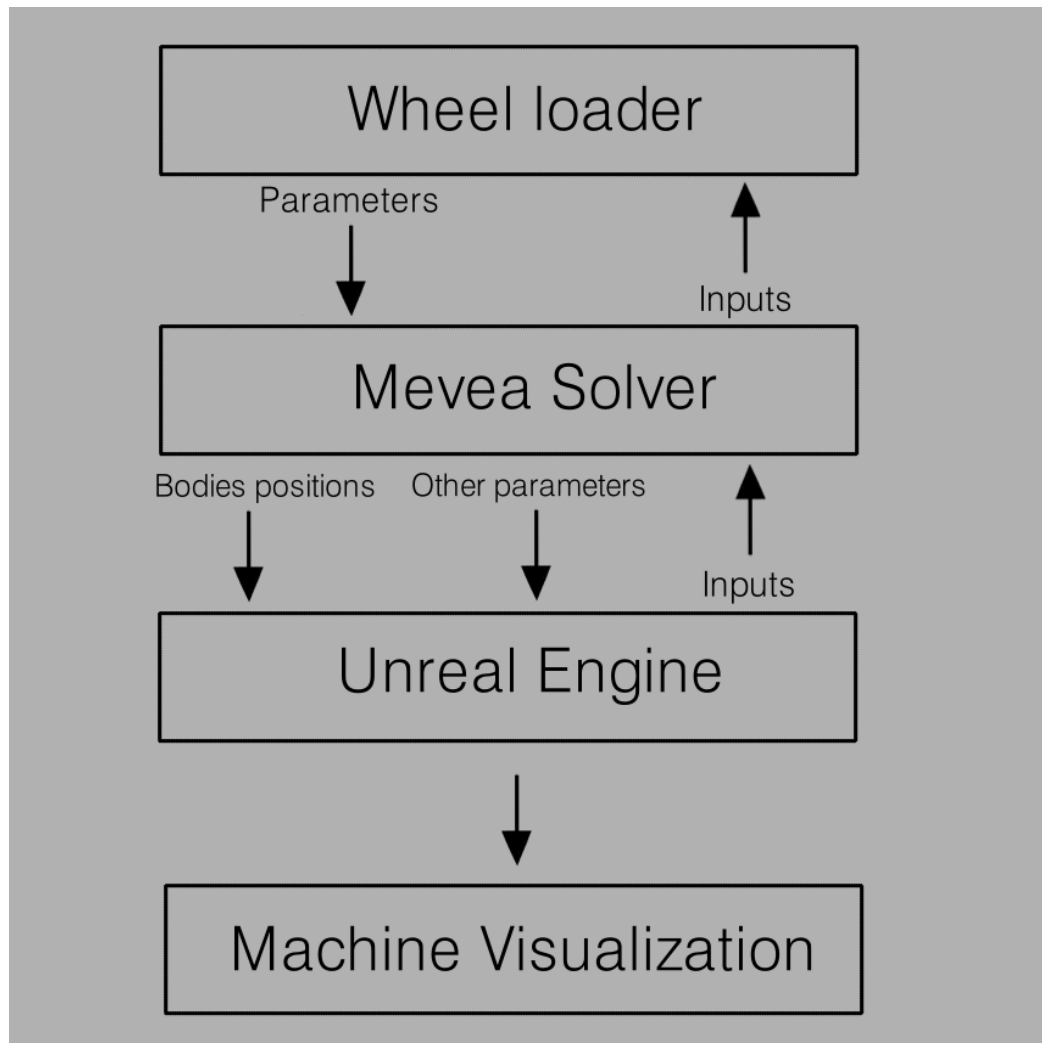


Figure 18: Data exchange between applications

During the simulation, the user is communicating with UE4 and don't have any access to Mevea Solver. Inputs signals, which are obtained from the steering wheel, are processed by UE4 and stored as local parameters inside of the specially designed C++ class. Then, these local parameters are packed and sent by the socket to Mevea Solver, where they are unpacked and assigned to the inputs of the model. On the next step, Mevea Solver calculates the machine movement according to specific inputs and sending the data about the model statement back to UE4. The data is sent in the real-time with the application of the UDP IP protocol, which is sending datagrams from one application to another.

For a better understanding of the protocol and transport layer choice, we must define the requirements, which are applied in this work for data transfer. The data must be sent in the real-time. Therefore data must be sent as fast as it is possible. This requirement is created

mostly by the application of the head mounted display, where small lags during visualization are causing virtual reality thickness. Of course, it is more preferably to have reliable data transfer, which can handle possible packets loss during transmission. At the start of the work, it was decided to use the TCP transport layer, because it is guarantee packet receiving and Mevea has an interface for handling TCP protocol. The TCP layer has numerous advantages such as connection based approach, the data is automatically breaking up into packets, flow control and easy in implementation [14]. Nevertheless, considering this advantages, the TCP layer is not applicable to this work. The reason for this is the socket's waiting functionality, which waits till enough data is buffered up before sending and resends the lost packets [14]. All these actions take much time and make the data sending process slower. The other transport layer, which is used by TCP IP is UDP. UDP stands for User Datagram Protocol, which sending packets without error checking or flow control. This type of communication allows to send the information as soon as possible, but it not guarantee that packets will be delivered. Packets can be lost due to multiple reasons, for example during sending through the internet (it is possible that 1-5% of the packet will be lost during transmission) or while the receiving side is not listening to socket. However, despite the unreliability of the UDP layer, it is preferable for the current application. The reason for this decision is the fact, that insignificant packet loss is preferable for real-time application than constant delays in the data transmission.

As it was mentioned before, a UDP socket interface requires sending and receiving clients for data transfer. According to the previously mentioned reasons, the application is sending a limited amount of the data via socket. This is the reason for creating multiple sockets, which are transferring certain data. At the Mevea Solver side, there are four sending clients. They are handling data, which is shown at figure 19. Namely, parameters are related to bodies and their corresponding parameters, which are sent by socket interface.

For each of the bodies, which are listed on figure 19, assigned Python scripts. These scripts are including socket and struct modules. Inside of the scripts, by default are two functions: `initScript` and `callScript`. The `initScript` function executes code when simulation started. The `callScript` function is similar to the tick function in UE4, which executes code every change of the frame.

The `initScript` function executes code, which is related to the start of the UDP socket. At this stage, the program defines valuable information about socket. It is defining the address which should be taken. The address includes the IP address and port number. For the sockets, which are used in this work were used 127.0.0.1 IP address and ports from 10000 to 10005. After defining the address, buffer size is defined. For the work of sockets, which are transferring data about bodies and dummies chosen the buffer size is 28 bytes. For the socket, which sending model parameters used the buffer size of 20 bytes. The reason for this decision will be explained below. After defining socket parameters, a socket is initialized. On the next step, code call `GSolver.getParameter()` function for further reference in the code. The `getParameter` function is searching for a parameter through all existing parameters of the model and getting its value.

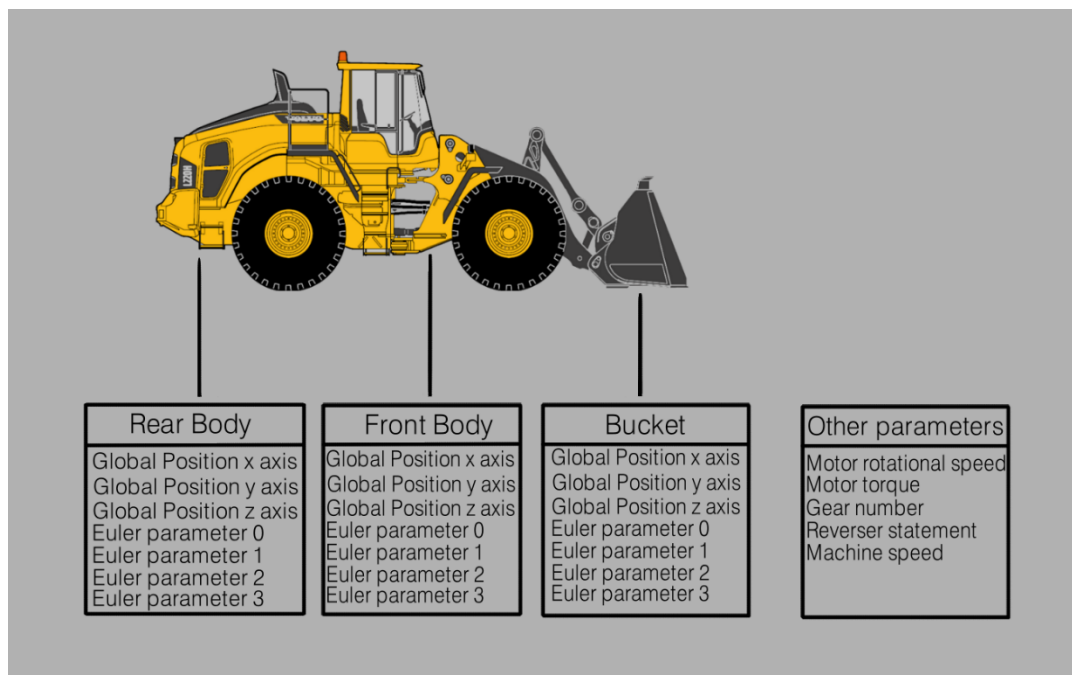


Figure 19: Parameters, sent by Mevea Solver application

The `callScript` function is sending seven parameters and receiving one parameter. At first step, function receiving delta time, which was obtained from UE4 and assign this value for Mevea Solver delta time parameter. This step was made for synchronizing two applications and was discussed previously. On the next step, function obtaining values of the body and dummies. Function obtaining global positions of a body in three axes and four Euler parameters. All parameters are floating point number. Therefore they are taking four bytes

of the memory. On the next step, these values are packed into the struct, which takes 28 bytes of memory. After packing of the data, a struct is sent.

In the UE4 is used only one sending socket, which is handling user-specified input parameters. For the particular work are used Logitech G29 Driving Force steering wheel for controlling machine movement and two Logitech Extreme 3D Pro joysticks for controlling hydraulic components. The steering wheel is shown in figure 20.



Figure 20: Logitech G29 steering wheel

The signal from the steering wheel and joysticks are gathered by UE4 and stored as local parameters in C++ class. Steering wheel consists of a wheel and pedals block. The steering wheel has steering input, paddle shifters and numerous buttons. The pedal block has clutch, brake and accelerator pedals. From all of the inputs equipped in this work are used steering for controlling position, paddle shifters for choosing gear, clutch, brake and accelerator pedals. Steering, clutch and brake inputs are represented by floating point value ranging from -1.0 to 1.0 and taking 12 bytes of the computer memory. Paddle shifters are represented by Boolean variable type and taking 2 bytes of memory for gear up and gear down signals. In overall, this socket sending 14 bytes of data via a socket interface. The socket working in the same manner as described before, packing variable into a struct and sending it to Mevea.

In this work were used five receiving sockets. Four sockets are created in the UE4 application and one in the Mevea application. They have relatively the same working principle regarding initializing of the socket. Each receiving clients are listening to the corresponding port and receiving datagrams. The struct from Python 2.7 has the same structure as a C++ struct. Therefore it does not require any additional steps to unpack it. After receiving translation and rotation of bodies, they are assigned to the corresponding actor. The SetActorLocation function is assigning a global position in three axes, which are similar to Mevea. The rotation of a body was challenging because global orientation value was limited to 90 degrees in the Mevea Solver. Therefore, were used Euler parameters, which can be used as quaternions (Shabana, 2001). The working principle of the receiving clients is shown in figure 21.

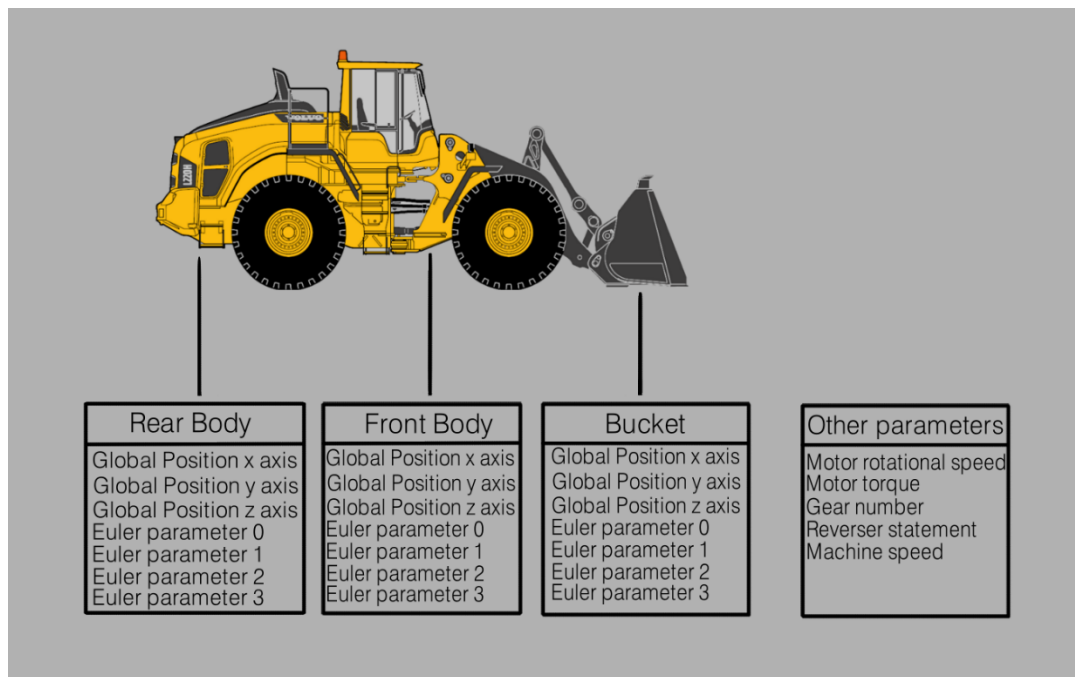


Figure 21: UE4 receiving clients operation principle

Another receiving client in UE4 is obtaining the parameters of the model. These parameters are used in the user interface, which shows the current state of the model during the real-time simulation. The socket is initialized, receiving the same struct as in other clients, unpacking it and storing parameters as local parameters. On the next step, these parameters are appearing as graphics instances inside the simulation. For user are displayed parameters related to motor, hydraulic system, gearbox and load. Parameters are listed in table 7 at the socket received parameters section.

The in-game user interface has three major functions. The first function is to show the current state of the machine for a user. The parameters, which are listed at table 7 under socket received parameters row, are going to be displayed in two different ways. The rotational speed of the motor and machine speed parameters, which are usually displayed on the dashboard of the machine, will be assigned to the dashboard actor. They are made as interface widgets, which showing only values and located above the dashboard mesh. The other parameters, listed below the socket received parameters row will be displayed on the dashboard as well. These parameters' indication differs in the displaying method. On the original machine's dashboard, gear number and reverse gear indication are made by indication lamps. That is why, these parameters are made by dynamic emissive material, which is emitting light under certain conditions. The motor torque and the pressure in a hydraulic system are displayed on the special monitor mesh in the same manner as machine speed and the rotational speed of the motor on the dashboard.

The second function is to show parameters, which are related to the visualization and interaction. These parameters are showing user's performance during the simulation and giving assistance features. User's performance information is represented by the number of collisions during the task, task completion state and points, which user has gained. These parameters are displayed in front of the user because they are connected with the HMD position. The headlights and fog lights control are performed by dynamic materials and are activated when the button mesh in the cabin has been pressed.

Table 7. *Parameters used in the in-game user interface.*

#	Name	Machine Part	Range	Units
Socket received parameters				
1	Rotational speed	Motor	Idle speed ...628	Rad/s
2	Motor torque	Motor	0...750	Nm

Table 7 continues. *Parameters used in the in-game user interface.*

#	Name	Machine Part	Range	Units
3	Gear number	Gearbox	-6...6	-
4	Reverse gear indication	Gearbox	0...1	-
5	Machine speed	General	0-90	Km/h
Parameters inside UE4				
6	Headlights on/off	Lighting	0...1	-
7	Fog lights on/off	Lighting	0...1	-
8	Collisions number during task	Gameplay	0...100	-
9	Task completion state	Gameplay	0...100	%
10	Points	Gameplay	0...100	-

The third function is control of the simulation state. This feature is controlling game base inside of the UE4. It is performed in the form of the menu interface. The menu actor is spawned in the world when the user hits the button, which is connected with the left mesh of the Leap Motion Hands pawn. The menu widget has three buttons. The first button is restarting the simulation by activating the code, which is closing the simulation file and opening it again. The second button stops the simulation and returns the user to the main menu.

One of the most important parts of the simulation is graphics. In general, it consists of 3D meshes, shader materials, lighting and post-process effects. Meshes are 3D models, made by special modeling software such as Maya, Blender or 3ds Max. They are representing only the geometry of an object. To these meshes are applied shader materials, which control the visual look of the surface, light reflection and casting of the shadows. In the world placed lighting sources, which are imitating real life light sources.

All 3D meshes can be classified as machine and world meshes. For this work was purchased 3D mesh of Volvo L220H wheel loader, which is shown in figure 22. The mesh consists of 805 000 polygons and 810 000 vertices. This fact shows that this mesh is applicable in the close-up rendering. Initially, this mesh has the max2009 format, but in the pack were

included six more files with different extensions. The pack also included textures, namely, base color, bump, gloss and opacity maps.



Figure 22: 3ds Max render the result

The model includes sixteen materials, which are shown in figure 22. Each of the material is assigned to the corresponding mesh. Materials under number 1, 11, 13 and 14 create a visualization of the plastic material, which is used on the outer parts of the machine. The material number two creates the base color of the machine clear coat. The material, which is specified by number four, creates the visual representation of the clear coat of cylinders, jointing bars and bucket. These materials are basic, which were made by applying a base color and roughness parameters. In other words, they have parameters showing their color in RGB and gloss of the surface. Materials under numbers 3, 6, 10, 15, are creating headlights' glass. They include a base color, bump, roughness and can pass outgoing light. Materials under numbers 5 and 16 have the base color, normal maps and roughness. Number 14 and 7 materials are creating glass and mirrors, that is why they have the base color, metallic and roughness parameters. The material number 12 is applied to the fire extinguisher mesh, which has a texture for the base color and roughness. Materials also depend on the UV mapping of the mesh, which is used for assigning textures to the 3D mesh.

Despite the fact, that model has good render results in the 3ds Max, the application of it in the UE4 was problematic and caused multiple errors during baking of the lighting. In figure

23 can be observed that the model has multiple shading artifacts, in other words, the problem with shadow casting. Also, on the glass of the windshield is displayed sign “invalid lightmap settings”, which indicating missing second set of the UV map.



Figure 23: Shading artifacts on the original mesh, the front plane

This problem was caused by the wrong layout of the UV map. The UV maps of the meshes were intersecting from 25-80 %. The example of the wrong UV layout is shown in figure 25 on the left side of the picture. The intersection of the UV maps is causing these artifacts because the shade is applied to multiple sides of the mesh. The correct example of the UV mapping is shown on the right of figure 24.

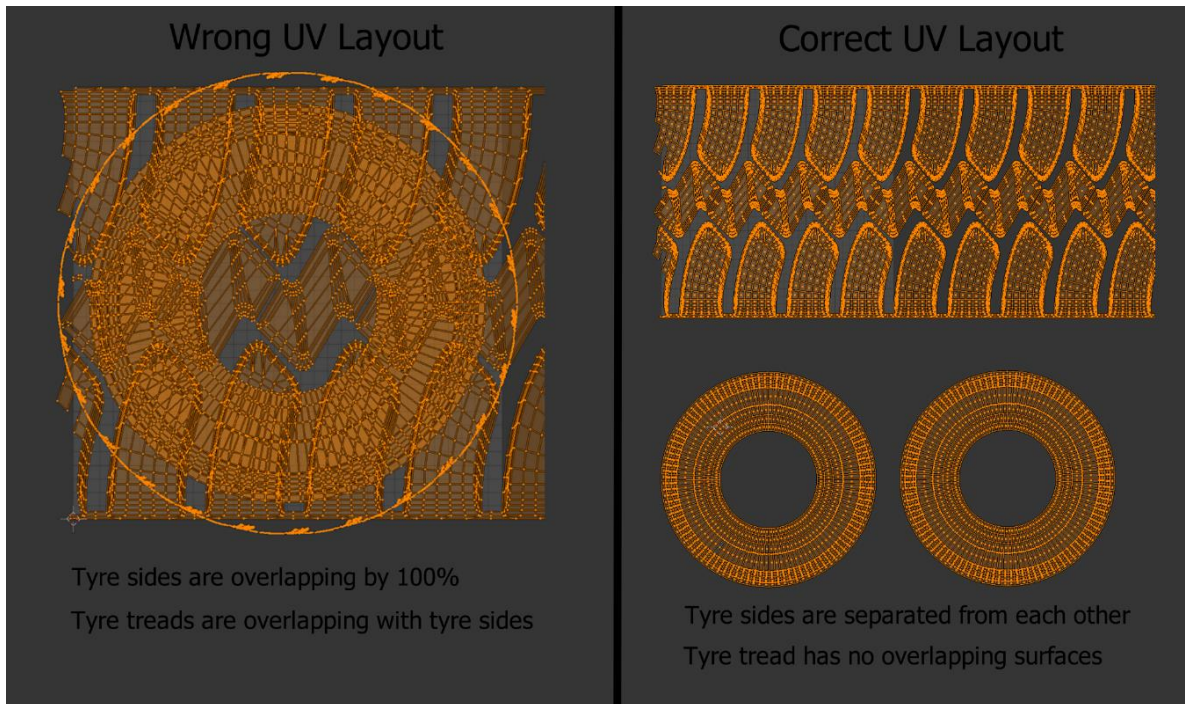


Figure 24: The wrong and correct UV layout example.

For fixing of this issue, the model was imported to Blender 3D modeling software. For each part of the 3D model were created new UV layouts, which has no intersection between their parts. The model was exported as fbx file and imported to the UE4. As a result, all the artifacts, which were present in the previous model version were eliminated. The model was placed in the created world map, which is shown in the figures 25 and 26.

The second part, which is related to the graphics of the application is world visualization. It consists of meshes, lighting and post-processing effects. This map includes eleven meshes, which are representing mountains, foliage and sky. The mountains meshes are divided into two groups: close and long-range meshes. This grouping is made for the reduction of needed GPU power. At the close range, objects have materials with high resolution, because it is closer to a user's viewpoint. As long, as small details cannot be observed at far distances,

for the long range objects meshes are applied lower graphics settings. The same rule is applied to the foliage application because the foliage is expensive regarding the load to GPU.



Figure 25: Main menu render the result, back view

The applied lighting techniques are also crucial regarding GPU load. For the reduction of the load to the system was used lightmass importance volume feature. This feature controls emitted photons from the lightmass and mostly concentrate them on the chosen area. This feature reduces the time needed to lighten the scene when the game is started and reduces the load to the GPU by reducing the number of photon bounces. The light source of the scene is a static directional light, which precomputes shadows in the scene and reduces the load to the system as well.



Figure 26: Main menu render result, front view

3.2 Building hardware for the user interface

During the simulation process, two applications are running simultaneously. These applications use different types of hardware and produce a higher load to the system. Mevea software is calculating dynamics of the machine, which including all the parts related to the hydraulics, collisions, parts like the motor and the transmission. This kind of computations are performed at the real-time, that is why load to the processor and RAM is high. At the same time, UE4 is calculating the graphical part of the model and displays it on the HMD in real-time, which requires high resolution of the scene and parts of the machine. That is why UE4 is creating a high load to the GPU of the computer. Generally speaking, during the operation of the simulation all the systems of the computer are highly loaded. For meeting the requirement of the efficient and correct work of the simulation, there is a strong need in high-performance hardware, which can withstand such kind of load.

The target computer, which is used for performing the simulation is mounted inside the simulation studio. The simulation studio was described in chapter 3. The computer specification is shown in table 7. In front of the corresponding part of the computer is specified applicability of it for the particular case. According to the specification, computer meeting requirements for the CPU, RAM and output devices, such as HDMI and USB ports.

Nevertheless, the computer is not meeting the requirement for the GPU, because of the increased load regarding graphics.

Table 8: *Simulation studio specification.*

Hardware	Specification	State
GPU	NVIDIA GTX 980	Not applicable
CPU	Intel i7-8650	Applicable
RAM	16GB	Applicable
Video output	Compatible HDMI 1.3 video output	Applicable
USB ports	3x USB 3.0 ports plus 1x USB 2.0 port	Applicable
OS	Windows 7	Not Applicable

For ensuring that the computer will be able to withstand the load created by applications, it must have a higher specification. According to the table 7, there is a need to the upgrade of the computer. The GPU of the target computer must be changed to the newer version. The researcher proposes to replace the GPU to one of the following options: Nvidia Geforce GTX 1060, GTX 1070 or GTX 1070 Ti.

3.3 Management data obtained

During stages of application work, a massive amount of the information is collected, which will be used in the subsequent development process. In order to make this data applicable to the analysis, it must be stored, grouped and reported. The following requirements were set to the data collection:

- Automate the process – the data must be handled automatically;
- Easy to handle – the data must be stored in the easy to read format, for example, an Excel file or as txt extension file;

During the single simulation sequence, the application collects different data in three steps. The first step is performed at the start of the simulation when a user started the application

and is in the main menu. This is the most important part of the application because a user states himself/herself and choose the initial parameters of the machine. The next step of the data collection procedure does not involve a user, because at this stage is collected data about machine behavior. The third step is related to the user feedback, which defines the user satisfaction rate about model behavior under chosen parameters or presets.

There is a clear need in the grouping of the data obtained during the application operation. The overall data must be divided into two groups in order to make clear reporting of the results. The first group is related to the user and the machine parameters information, which will be collected into a single csv file. The other group of information is related to the application performance, which is needed for the maintenance and debugging of the application in future. This information is stored in a simple txt file, which writing all errors and warning during the application operation. This kind of grouping of the data is made for separating the information from different team members and will be described in chapter 3.7.

3.4 The main menu obtained data

This part of the data is collected by Leap Motion user interface in the main menu. The first part of it is related to the user demographical data. The second part of the information obtained is user-defined parameters settings of the machine. This part of the information is crucial because it is used as initial parameters to the simulation.

3.4.1 User information data

The user information data includes basic information about the user. It is asked for every new sequence of the simulation, therefore for every time simulation was started again, the user profile is defined. The user profile includes information about age, sex, occupation and preferred language of the interface.

The application asks for the user information when the start simulation button is pressed. The questioning form is made as the widget with combo boxes, radio buttons and buttons, which are used for choosing needed options. The combo box is used for obtaining the age of a user because it includes large amount of options. The age options have ranged from ten to one hundred years. Language, age, sex and occupation questioning is made by combo boxes,

as long as there is a small amount of possible options. The possible options for occupation are management, engineering, educational, IT, student and other. The user information is important and closely related to the machine initial parameters data because it is crucial for the filtering of the parameters sample.

3.4.2 Machine initial parameters

As it was described previously, initial parameters obtained at the main menu are used for filling up the machine parameters XML file. These parameters include information about bodies mass, motor and transmission of the machine. They are part of the first CSV file and located on the same sheet as user-defined data.

3.5 Data obtained during the simulation

The next part of the data sample is obtained during the real-time simulation of the machine. These parameters are displaying the state of the machine during the simulation procedure under chosen initial parameters. The storing of the parameters values are bonded to the tick function, in other words, on each change of the frame. Therefore, the amount of values, which must be stored in each simulation is massive. That is the reason for saving this data on the separate CSV file. This data sample includes the general state of the machine, forces values, motor and power transmission parameters. The list of the stored parameters is shown at table 7.

Table 9: *Machine statement stored parameters.*

#	Name of the parameter	Type
1	Machine speed	General
2	Left Cylinder Lift, a translational force	Forces
3	Right Cylinder Lift, a translational force	Forces
4	Left Cylinder Tilt, a translational force	Forces
5	Right Cylinder Tilt, a translational force	Forces
6	Rotational speed	Motor
7	Motor torque	Motor
8	Torque converter output torque	Power transmission
9	Torque converter output angular velocity	Power transmission

Table 9 continues. *Machine statement stored parameters.*

#	Name of the parameter	Type
10	Gearbox output torque	Power transmission
11	Gearbox output angular velocity	Power transmission

3.6 Data obtained after the simulation

After finishing the simulation and storing all the previous parameters, there is a need for the next stage of the data collection procedure. During this stage is collected information about the user satisfaction rate. The satisfaction rate information includes an opinion about the current configuration of the machine and opinion about the application. This data is made in the form of the simple answers to the set of questions. During this stage asked the following questions:

- Do the motor have enough power? **Answer options:** yes/no.
- Is the idle motor speed enough to start a movement of the machine? **Answer options:** yes/no.
- Should the idle speed be increased or decreased? **Answer options:** increased, decreased, keep these settings.
- Should the gear ratio be increased or decreased? (for each gear) **Answer options:** increased, decreased, keep this setting.
- Which gear is the most suitable for handling the load on the short distance? **Answer options:** gear 1, gear 2.
- Which gear is the most suitable for handling the load on the long distance? **Answer options:** gear 3, gear 4.
- Describe the application operation. Was the picture freezing or displayed precipitously? **Answer options:** yes, the picture was freezing all the time; yes, the picture was sometimes freezing; no, the picture was displayed correctly.
- Describe your comfort level. Did you feel sick during the simulation? **Answer options:** yes, heavy dizziness; yes, slight dizziness; no, the experience was comfortable.
- Please, rate the application in scale from 1 to 5. **Answer options:** options from 1 to 5 stars.

3.7 Output files structure

As it was discussed earlier, the results of data collection will be divided into three main files categories: two CSV and two TXT file. The first csv file type is containing data about the user, his/her defined machine parameters, opinion about the current configuration of the machine and application performance data. The second csv file type containing the information about machine statement during the simulation sequence. The TXT file contains the information regarding exceptions and errors during work of the application.

Storing the information about each simulation sequence is important. This is the reason a for binding files to simulation sequences. The data is stored in CSV files by the number of the corresponding simulation sequence. The application is assigning the unique number to each sequence. The data obtained at the particular sequence is displayed under the unique sequence number. By searching the needed number of the sequence in the output files, can be accessed any data, which is needed for a development team.

Structure of the first CSV file is shown in the Appendix I. This is the representation of the structure of the file and does not have the same structure. The first row is specifying the number of the simulation sequence. After simulation n sequence number follows the user data block, which consists of age, sex, language and occupation. Most of the rows in the user data block are using abbreviations and short words with length less than six digits. This designation helps to avoid confusion during subsequent data analysis. The next block contains the information of the machine parameters, such as a motor, transmission, mass properties of parts and presets options. The preset option rows display the state option as Boolean. This option is showing if a user chose to apply presets. If a user specified using of preset, it is marked as one, otherwise, zero. The second row displays some chosen preset by their order in the application. The next block displays the data about the user satisfaction rate. The row name is showing a number of the question as they are ordered in the application. Questions, which has two types of answer, the output displayed by zero and one. These digits display the number of an answer. In a question with two types of answer zero stands for yes and one for no. In the questions, which has three types of answer, options are displayed from zero to three. In the same manner, in the question with five types of answer, which specifying customer satisfaction rate about application options are displayed from

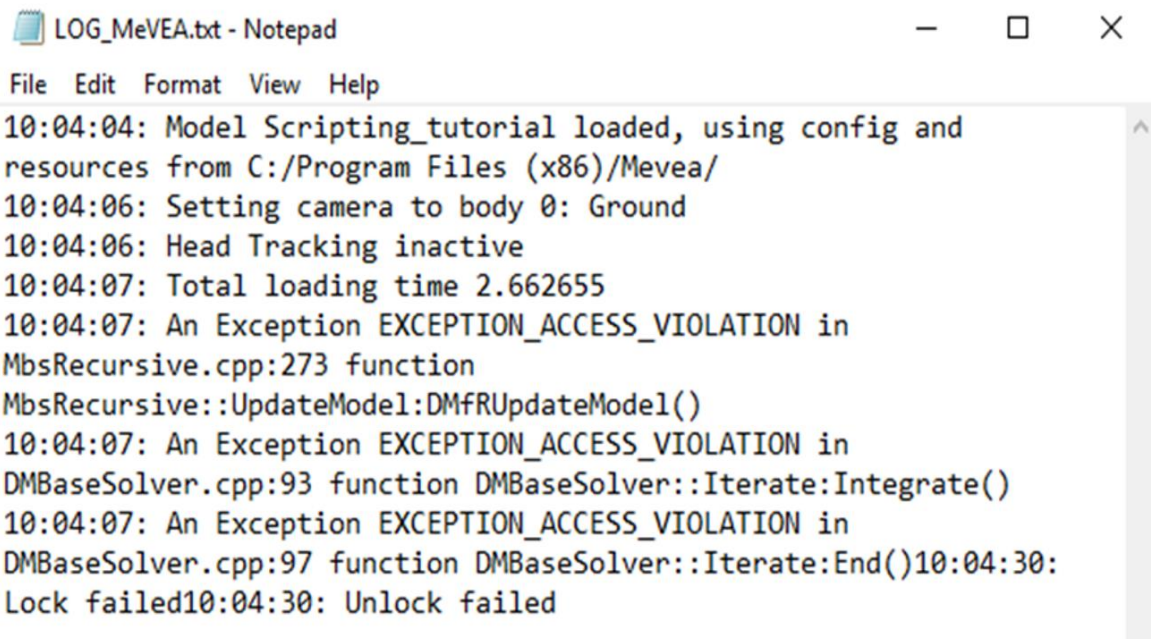
zero to four. For avoiding confusion with designations, the file contains legends for all types of data listed.

This kind of structure gives the advantage to further data analysis. This file contains all the needed information regarding the user and chosen configuration. The data structured in one table. This fact creates an advantage for the filtering of the data according to the user's group and specified option.

The second type of the files is the generated file during the simulation. As long as CSV file does not support the format with multiple sheets, each simulation sequence creates its CSV file. The simulation output file has the name, which corresponding to the number of the simulation sequence. It contains information about the machine state during the simulation. The information is displaying parameters of the motor and the transmission, namely motor rotational speed, motor torque, gear number used and the reverser statement.

The simulation sequence lasts for three hundred seconds. During the simulation, the application is collecting data at each second. As the output, produced three hundred values for each parameter. This kind of data gives the opportunity to define how the machine was used during the simulation sequence.

The next type of files, which is generated during work of the application is TXT file. These files contain log messages of Mevea and Unreal Engine 4 (UE4). Messages are displaying the state of the application, including warnings, exceptions and errors during work of the application. Mevea log text file is shown in figure 27. It is automatically generated file by Mevea Solver application. This TXT file is updated after each simulation sequence. That is the reason to save it in the separate directory. At each simulation sequence, this file is processed by UE4 and saved by the unique number of the sequence. The file contains time when some event occurs and a short description of the event. As it is shown in the example file, after loading of the simulation multiple exceptions related to the numeric solver are occurred. These files create an advantage in the long term application using. They give the possibility to track application and model performance during each simulation sequence.



```

LOG_MeVEA.txt - Notepad
File Edit Format View Help
10:04:04: Model Scripting_tutorial loaded, using config and
resources from C:/Program Files (x86)/Mevea/
10:04:06: Setting camera to body 0: Ground
10:04:06: Head Tracking inactive
10:04:07: Total loading time 2.662655
10:04:07: An Exception EXCEPTION_ACCESS_VIOLATION in
MbsRecursive.cpp:273 function
MbsRecursive::UpdateModel:DMFRUpdateModel()
10:04:07: An Exception EXCEPTION_ACCESS_VIOLATION in
DMBaseSolver.cpp:93 function DMBaseSolver::Iterate:Integrate()
10:04:07: An Exception EXCEPTION_ACCESS_VIOLATION in
DMBaseSolver.cpp:97 function DMBaseSolver::Iterate:End()10:04:30:
Lock failed10:04:30: Unlock failed

```

Figure 27: Mevea LOG file example

The second type of the TXT generated file is related to UE4. These files are containing information about the work of the UE4 application during each simulation sequence. They are reporting about warnings, exceptions and error in UE4 application. As long as UE4 is not closed after each simulation sequence and remaining in standby mode, files are generated at the end of the simulation. A file has the same naming system as Mevea simulation txt files, which are bound to the unique number of the sequence.

All files, generated during the application operation, are located in a system at the same directory, where the main simulation files are. This kind of layout is created due to the application of the relative path to the file when files are generated and saved. It creates an advantage in handling the information, especially, for sending and maintenance of the system.

3.8 Data sending

The next stage of data handling is sending files to the development and maintenance team for further processing of it. Generated files, which were described before, can be sent by two ways: by uploading them to cloud service or sending files via e-mail to a server. Depending on the case and type of file should be used different ways of data sending.

The cloud services are relatively easy to use for storing and transferring information between machines. For these purposes can be used for cloud services such as One Drive from Microsoft Corp. or Google Drive from Google Corp. In order to obtain data from the cloud service, two preparation operations must be performed. At first, should be created an account in the cloud service, which only handles the storage of the obtained data. At the second step, on the computer must be installed cloud service software, which allows saving files without accessing a browser. After finishing these steps, it is possible to save files into the cloud service. Files are stored at the computer directory first and after saved to the drive by UE4 application. Saving files to the cloud services have advantages to the case. It is simplifying finding and accessing the process for the particular file. Also, cloud service can be used as backup storage for the case of data loss. On the other hand, cloud services have a limit to the storage of the data. For Google Drive storage space is limited to 17 GB of the information and One Drive is limited to 5 GB.

As an alternative to the cloud service can be used e-mail notifications with attached generated files to it. At the first stage, the UE4 application is saving the file to the computer in the project folder. Then, the code in UE4 application sends files to the particular e-mail address via TCP socket. The advantage of this kind of data transfer procedure is a constant notification of teams about the statement of the machine and report about collected data. As the drawback of this sending procedure is the relatively small amount of data, which can be sent. It is not efficient to send all the generated files by e-mail.

According to the advantages and disadvantages listed above must be concluded as a way of the most efficient data transfer type. As long as cloud services can store big amount of data, it is preferable to save large files to the drive. To the cloud service stored files generated during the simulation sequence. They are machine parameters during the simulation, Mevea and UE4 log files. On the other hand, it will be effective to send the initial parameters file via e-mail for the effective observation of the study results.

After the finish of the application work, received study results in the generated files can be used in the development process. This data is describing the user behavior regarding configuration and user satisfaction rate about it. This kind of data gives an opportunity to create a weighted decision about future machine design according to the user preferences,

legislation and standards applied. Nevertheless, the information in the generated files is raw data, which requires proper filtering and sampling.

3.9 Data filtering

Depending on the environment, where the simulation studio is installed, the resulting data sample can be different. In some cases, for example, conferences or trade fair related to the heavy machinery equipment may produce more valuable data set in comparison with other random locations, such as shopping malls, because there is a higher concentration of the potential customers. That is the reason for data filtering before the actual application of the results.

The main objective of the data filtering is to define the preferred group, which can make weighted decisions about the parametrization of the machine. The overall sample can be divided into three main groups. The first group is representing the sample, which including the most valuable users. As the most valuable users are an implied person, which are working with that type of machines or has been working with them before. The second group represents intermediate importance users, which are in the management positions related to the heavy machinery industry or industries, where this kind of machines is involved. The third group is representing insignificant users, which do not have any experience or relationship to the heavy machinery or adjacent industries.

Generally, the filtering procedure requires two major steps, which are shown in figure 28. The first step is made by VBA code in MS Excel. Whole data is filtered by code, which sorts information according to the certain criteria. At this step, data is filtered according to the age and occupation criteria. As it was mentioned before, there are three major classification groups. The age group under eighteen years is classified as insignificant for the development purposes. The age group including users from eighteen to twenty-five is classified as intermediate importance group and can be partially used in the decision-making process. The most important group is including users from twenty-five to one hundred. As it was discussed before, occupation classification also consists of the same three groups. After finishing the first step of filtering data is ready for further processing.

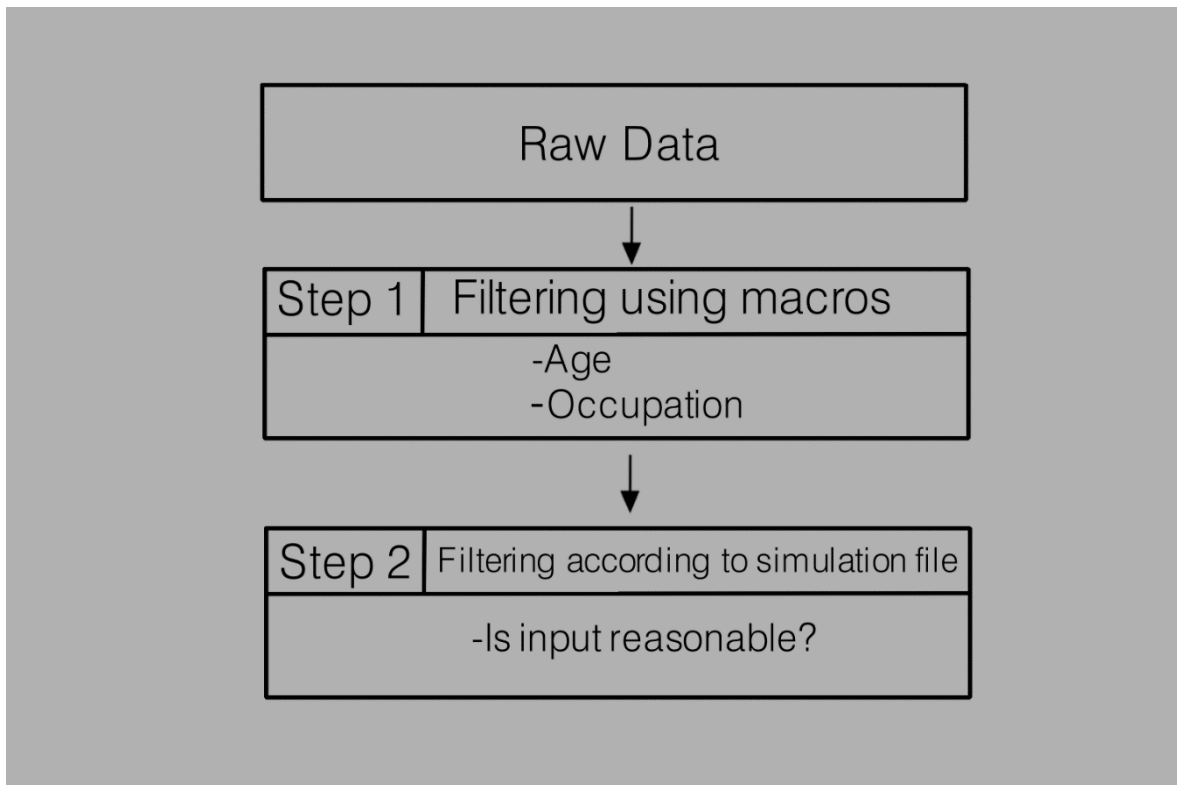


Figure 28: Data filtering major steps

At the next step, data is filtered according to the justification of the user. In this step is involved the file, which containing the information obtained during the simulation sequence. This step is performed by a development team. The main objective is to define the behavior of the user during the simulation. The sample, which was developed during the first step of the filtering is used for analyzing. From this sample are collected numbers of the sequence, where the user evaluated machine behavior as good. Machine behavior assessed as good when the average satisfaction rate is higher than three points out of five. These simulation sequence numbers are extracted from the overall sample. According to the initial parameters file, each sequence is analyzed by the level of configuration credibility. In other words, the development team justifies if the configuration created by a user is plausible and adequate in the real world application. During the second stage of the data, filtering is used parameters obtained during simulation as well. By the file, which tracks the state of the machine during the simulation, can be analyzed how the machine was used. According to the rotational speed of the motor, torque, gear number and reverser status can be analyzed the behavior of the user. For example, it is possible to define if the user has collided with the object, or made some other mistakes during the simulation. The data is accepted for further development

after checking on state of machine. If no abuse or incorrect using detected, data is accepted for the concept creation stage.

4 ANALYSIS

After the filtering and accepting the simulation results, they are used in the development of the main concept of the machine. The procedure of the development process is shown in figure 29. Created concepts are evaluated regarding reliability and meeting standards applied to the machine, such as ecological and safety standards. The research team approximately evaluate the lifetime of the machine components, possible emissions, environmental and safety risks. The most optimal solutions, which are meeting the requirements regarding reliability and meeting standards are passed to the next stage, where estimated manufacturing costs for each concept are defined.

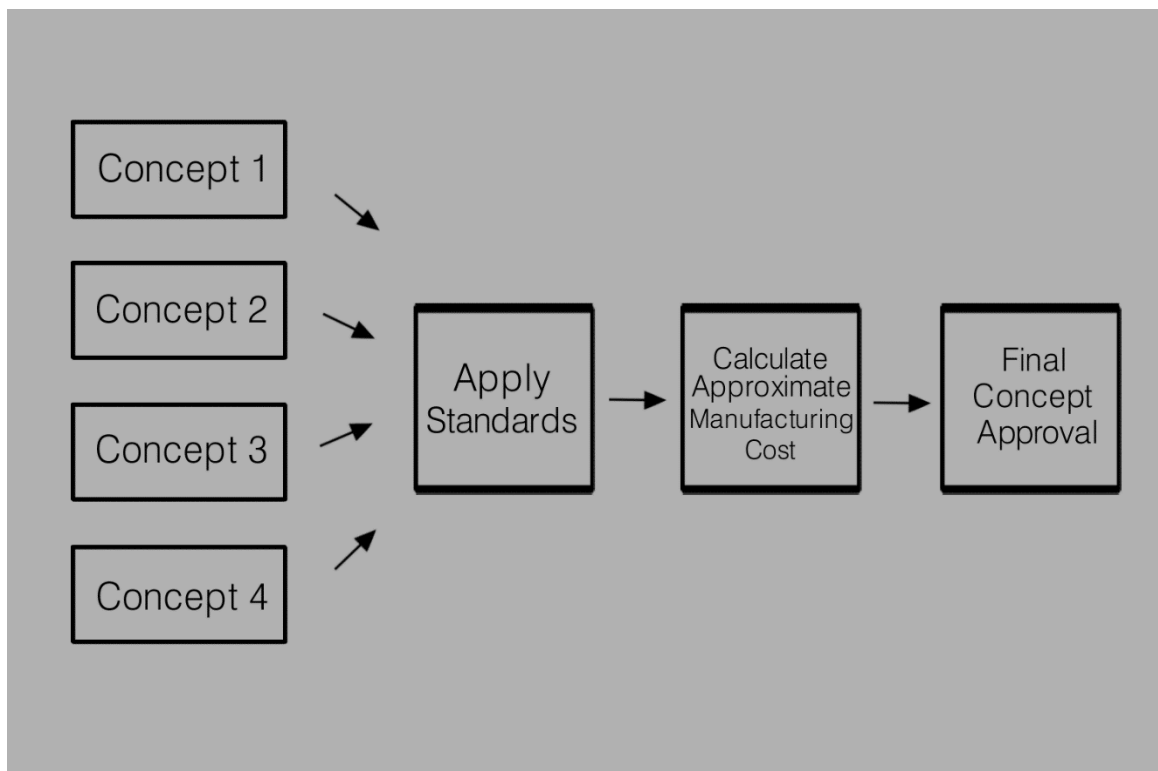


Figure 29: Application of the data in the development process

The approximation of the manufacturing costs starts with creating a preliminary list of the components needed for creating a machine with needed characteristics. After this step, the concept, which is the best regarding the manufacturing cost is defined as a final concept. The final concept is proceeding further for the detailed development and subsequent manufacturing.

4.1 Practical application example

The practical application of the data can be observed in the example of motor parameters. In this work were collected user-specified parameters of the motor such as the torque curve and idle speed of the motor. The collected data is sorted according to the user data, such as age and occupation. After filtering of data, will be collected the most popular setup of the motor among users. These setups are new concepts of the motor. The next step is involving the development team to analyze results and define the most efficient concept of the motor. During this stage, concepts are checked if they are satisfying standards and the approximate manufacturing cost is calculated. The concept, which passed all stages of the process is evaluated as a final concept.

4.2 Test group results

For the test of performance and functionality of the interface was used test group of four people. The test group was evaluating the performance and usability of the user interface. The respondent stated that application works with good performance. The respondent stated that during and after using application there was no sickness. Therefore, the comfort level of the VR experience is evaluated as good. There was following advice for application improvement:

- The in-game user interface must be connected to head mounted display rotation;
- The application must have the ability to reset the position of the headset at the start of simulation;

4.3 Achievement of research results

During the thesis work, the following result were achieved:

- Was created the application, which allows changing physical parameters of the real-time simulation model;
- The application can filter data and send it to the development team;
- The concept evaluation procedure was created, which applies simulation obtained data.

5 CONCLUSION

The main research objective was defining a method of efficient gathering of user feedback and converting it into the valuable data for the research team. As an outcome of the research was developed a VR application which collecting the user-defined simulation model's physical parameters. The software gives the possibility to collect data from users automatically and use it in the product development process.

From the research, problem was produced three research questions. The first question was related to the method and tools of the simulation model parametrization. The parametrization of the simulation model was performed by modification of the simulation parameters XML file with XML parser. The second question was focusing on the method of visualization. The visualization of model and interface was created by applying UE4 and custom UDP sockets for transferring model parameters. The third question was related to the collection and analysis of the obtained data. The data handling is divided into three parts. At the first part, data is filtered by the software according to the user information. At the second step, the filtered data is written into a CSV file, which is sent to the development team. At the third step, the research team evaluating the results and creating concepts by this information.

The application, which was created during this thesis, provide vast opportunities for the industrial companies and average users. Industrial companies are obtaining the technology, which gives the opportunity to develop a customer-oriented products with higher efficiency. The products, which may be more appealing to the customers, will be produced faster and with fewer costs on the development. At the same time, applications this kind may result in a better brand image, because it shows the customer, that their opinion is important for the manufacturer.

The customer receives a more appealing product. This reasoning supports factors of the price and customer oriented development. In the industry, the cost is one of the most important factors in the decision making about purchasing the machine. This is the reason, that customer may choose a cheaper product with good quality. Under the quality are applied

important merits, such as reliability, performance, usability, user-friendliness and ergonomics of the product.

This work creates a room for the further development of this topic. For the further studies are suggested application of the artificial intelligence for the analyzing of the results of the studies and development of the machine concepts.

LIST OF REFERENCES

- [1] R. Schmitt, B. Falk, S. Stiller and V. Heinrichs, "Human Factors in Product Development and Design", *Lecture Notes in Production Engineering*, pp. 201-211, 2014.
- [2] J. Gao and A. Bernard, "An overview of knowledge sharing in new product development", *The International Journal of Advanced Manufacturing Technology*, vol. 94, no. 5-8, pp. 1545-1550, 2017.
- [3] D. Brabham, "Crowdsourcing", MIT Press, pp. 2-59, 2013.
- [4] R. Stephens, "Beginning software engineering", Wrox, a Wiley brand, 2015.
- [5] "Extensible Markup Language (XML) 1.0 (Fifth Edition)", W3.org, 2018. [Online]. Available: <https://www.w3.org/TR/REC-xml/#dt-docent>. [Accessed: 22- Feb- 2018].
- [6] B. Joshi, "Beginning XML with C#". [S.l.]: APRESS, pp. 1-7, 2018.
- [7] A. Salminen and F. Tompa, *Communicating with XML*. Boston, MA: Springer Science+Business Media, pp. 1-119, LLC, 2011.
- [8] "Well Formed XML - w3resource", w3resource, 2018. [Online]. Available: <https://www.w3resource.com/xml/well-formed.php>. [Accessed: 22- Feb- 2018].
- [9] M. Ltd., "Mevea | Software and services to build your own digital twins", *Mevea*, 2018. [Online]. Available: <https://mevea.com/>. [Accessed: 06- Sep- 2018].
- [10] M. Alger, "Visual Design Methods for Virtual Reality", 2015. [Online]. Available: <https://drive.google.com/file/d/0B19I7cJ7tVJyRkpUM0hVYmxJQ0k/view>. [Accessed: 27- Feb- 2018]
- [11] "Oculus Rift | Oculus", *Oculus.com*, 2018. [Online]. Available: <https://www.oculus.com/rift/>. [Accessed: 11- Aug- 2018].

[12]"Technology - Leap Motion", *Leap Motion*, 2018. [Online]. Available: <https://www.leapmotion.com/technology/>. [Accessed: 14- Aug- 2018].

[13] "Hardware and Software Specifications", *Docs.unrealengine.com*, 2018. [Online]. Available:<https://docs.unrealengine.com/en-us/GettingStarted/RecommendedSpecifications>. [Accessed: 15- Aug- 2018].

[14]"UDP vs. TCP | Gaffer On Games", *Gafferongames.com*, 2018. [Online]. Available: https://gafferongames.com/post/udp_vs_tcp/. [Accessed: 24- Jun- 2018].

APPENDIX I

Main output CSV file

	Data about user										Machine parameters										Satisfaction rate										
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	
1																															
2																															
3	Sequence	User age	User sex	Language	Occupatic	Point 1	Point 2	Point 3	Point 4	Point 5	Point 6	Motor idle	Gear ratios of the gearbox				Mass properties		Presets		01	02	03	04	05	06	07	08	09		
4	1	24	male	fr	m	100	200	450	725	550	480	62	3.8-4.15	1.9	0.95	0.5	1900	6850	250-350	yes	1	0	1	1	1	1	1	1	1	1	
5	2	18	female	en	e	120	254	500	650	750	400	75	4.1	2.1	1.1	0.75	1700	6000	300 no	yes	4	0	0	1	2	1	1	0	1	2	
6	3	16	male	en	e	-	-	-	-	-	-	-	-	-	-	-	-	-	-	yes	4	0	0	1	0	0	2	2	1	3	
7	4	35	female	en	m	-	-	-	-	-	-	-	-	-	-	-	-	-	-	yes	2	1	1	0	2	1	0	1	1	2	
8	5	25	female	fr	m	170	369	425	735	750	700	80	3.8	2	0.98	0.55	1900	6500	260 no	-	1	0	1	0	0	0	2	2	2		
9	6	28	female	fr	m	179	278	437	550	710	650	76	3.9	1.9	1.25	0.65	1800	5500	325 no	-	1	1	0	1	1	1	1	1	1	3	
10	7	27	male	fr	s	182	285	521	560	670	650	65	4.15	1.95	1.2	0.62	2000	5200	340 no	-	1	0	2	2	0	1	0	2	2	4	
11	8	35	male	en	it	177	354	502	680	632	684	62	4.15	2.15	1.15	0.5	2200	6600	315 no	-	0	1	2	2	2	0	1	2	1	4	
12	9	33	male	en	it	-	-	-	-	-	-	-	-	-	-	-	-	-	-	yes	3	1	0	0	0	1	0	2	2	3	
13	10	22	male	en	o	135	264	498	725	632	635	75	4.05	2.2	1	0.45	2100	7000	300 no	-	0	1	1	2	2	0	1	1	1	4	
14	11	15	male	fr	m	-	-	-	-	-	-	-	-	-	-	-	-	-	-	yes	4	0	0	2	2	0	1	2	2	2	
15	12	12	male	fr	e	-	-	-	-	-	-	-	-	-	-	-	-	-	-	yes	4	0	1	2	2	0	0	1	2	2	
16	13	23	male	fr	e	132	214	516	712	732	725	74	4	2.05	1.05	0.68	2150	6800	280 no	-	1	1	2	1	0	1	2	1	2	2	
17	14	29	male	en	m	176	267	546	745	741	755	72	3.8	2	1.2	0.7	2000	6250	250 no	-	1	1	0	0	0	0	2	2	2	2	
18	15	40	male	en	m	154	235	432	732	722	736	80	3.85	1.98	1.15	0.73	2300	5200	260 no	-	1	1	1	1	1	1	1	2	1	4	
19	16	35	male	en	m	123	287	462	723	715	763	65	3.9	1.95	1.17	0.64	2400	5500	270 no	-	1	0	1	0	0	1	2	2	2	4	
20	17	56	male	fr	s	-	-	-	-	-	-	-	-	-	-	-	-	-	-	yes	1	0	0	2	1	1	0	1	2	3	
21	18	34	female	fr	it	114	301	528	725	711	759	67	-	1.99	1.25	0.45	-	6000	320 no	-	0	0	1	1	1	1	1	2	1	4	
22	19	30	male	fr	it	-	-	-	-	-	-	-	-	-	-	-	-	-	-	yes	3	1	1	2	0	0	1	2	2	3	
23	20	21	male	en	o	-	-	-	-	-	-	-	-	-	-	-	-	-	-	yes	2	0	0	1	1	0	0	1	1	2	
24	21	20	male	en	m	164	395	475	723	700	698	71	3.9	2.1	1.25	0.58	2180	7000	275 no	-	1	1	2	1	0	1	2	2	3		
25	22	26	female	en	e	172	231	465	714	689	687	69	4.15	2.16	1.15	0.57	1950	5000	315 no	-	1	0	1	0	1	1	1	2	2	1	
26	23	14	male	fr	e	183	264	432	746	654	632	63	3.85	2.2	1	0.48	1850	5300	320 no	-	1	1	1	1	1	0	2	2	1	1	
27	24	17	male	fr	m	155	278	412	723	615	756	80	4.05	2.11	0.98	0.52	1900	5150	350 no	-	0	0	2	1	1	1	1	2	2	1	
28	25	19	female	fr	m	-	-	-	-	-	-	-	-	-	-	-	-	-	-	yes	2	1	0	2	0	1	1	2	2	2	
29	26	27	male	en	m	135	333	511	750	690	682	71	3.8	2	0.97	0.45	2000	6500	250 no	-	0	1	2	1	0	1	1	2	2	4	
30	27	38	male	en	s	-	-	-	-	-	-	-	-	-	-	-	-	-	-	yes	-	0	0	1	0	1	0	1	2	1	3
31	28	47	male	en	it	-	-	-	-	-	-	-	-	-	-	-	-	-	-	yes	1	0	1	0	1	0	1	1	1	4	
32	29	32	male	fr	it	158	327	534	715	658	625	68	3.85	2.08	1.25	0.7	2200	6800	265 no	-	1	0	1	0	0	1	0	2	2	4	
33	30	17	female	fr	o	134	365	432	746	694	613	75	4.05	2.1	1.1	0.73	2400	5300	340 no	-	0	0	1	1	1	1	1	2	2	4	