LAPPEENRANTA UNIVERSITY OF TECHNOLOGY

FACULTY OF TECHNOLOGY

THE DEPARTMENT OF MECHANICAL ENGINEERING

INTEGRATION OF REAL-TIME SIMULATOR, MOTION PLATFORM AND HAPTICS

Examiners: Professor, D.Sc. Heikki Handroos, Professor, D.Sc. Huapeng Wu
Supervisors: Professor, D.Sc. Heikki Handroos, Professor, D.Sc. Huapeng Wu

Jani Heikkinen
Lappeenranta 15.06.2009

ABSTRACT

Author: Jani Heikkinen

Title: Integration of Real-time simulator, motion platform and haptics
Department: Mechanical engineering
Year: 2009 Place: Lappeenranta
Thesis for the Degree of Master of Science in Technology.
65 pages and 46 figures.
Examiners: Professor, D.Sc. Heikki Handroos, Professor, D.Sc. Huapeng Wu

Keywords: Real time simulator environment, haptic device, motion platform, Virtools model

This thesis describes the process of the integration of a real-time simulator environment with a motion platform and a haptic device as a part of the Kvalive project. Several programs running on two computers were made to control the different devices of the environment. User tests were made to obtain information of needed improvements to make the simulator more realistic. Also new ideas for improving the simulator and directions of further research were obtained with the help of this research.

TIIVISTELMÄ

Tekijä: Jani Heikkinen

Nimi: Integration of Real-time simulator, motion platform and haptics

Koulutusohjelma: Konetekniikka

Vuosi: 2009 Paikka: Lappeenranta

Diplomityö. Lappeenrannan teknillinen yliopisto. Teknillinen tiedekunta.

65 sivua, 46 kuvaa.

Tarkastajat: Professori, D.Sc. Heikki Handroos ja Professori, D.Sc. Huapeng Wu

Hakusanat: reaaliaika simulaattori, haptiikka, Stewart liikealusta, Virtools malli

Tämä työ kuvaa projektia reaaliaikasimulaattorin rakentamisesta ja käyttöönotosta osana Kvalive projektia. Reaaliaikasimulaattoriin liitettiin haptiikkalaite ja liikealusta. Osana työtä suunniteltiin ja tehtiin useita ohjelmia yhdistämään järjestelmän komponentit toisiinsa. Näitä ohjelmia ajettiin kahdella tietokoneella. Lisäksi tehtiin käyttäjätestejä, joista saatiin lisätietoa siitä miten järjestelmää pitäisi kehittää, jotta siitä tulisi entistä realistisempi. Tämän projektin kautta saatiin myös ideoita simulaattorin parantamiseen ja jatkotutkimukseen.

ACKNOWLEDGEMENT

This master's thesis was carried out at the Mechanical Engineering Department of Lappeenranta University of Technology, Lappeenranta. I thank my professors Huapeng Wu and Heikki Handroos for giving me an opportunity to participate in an interesting research project. I also thank them for their very big help, comments and appropriate suggestions. My special thanks to MSc. Tatiana Minav for continuous support during this work. I also wish to thank my parents and my friends who supported me during my studies.

Lappeenranta, Finland, June 2009
Jani Heikkinen

TABLE OF CONTENTS

USED SYMBOLS AND ABBREVIATIONS

baudrate	Speed of serial transfer	[bits/s]
bytes	Number of bytes to transfer in one frame	[bytes]
bp	Bits per one byte	[bits]
sb	Amount of start bits	[bits]
eb	Amount of stop bits	[bits]

Abbreviations

VRPN	virtual reality peripheral network
BB	building block
DLL	dynamic link library
3D	3-dimensional
2D	2-dimensional
PC	personal computer
HMD	head mounted display
LUT	Lappeenranta University of Technology
DOF	degrees of freedom

1 INTRODUCTION

1.1 Overview of training and real-time simulators

Training simulators have already been used extensively for several decades in teaching people to use some device or machine without the need to use the actual system. Very often the real world systems are either too dangerous or too expensive to be used by a person just learning how to use it. Simulated systems have ranged from simulations of commercial investment environments thru military strategy simulators to car driving and shuttle flying simulators. In the Fig.1.1 there is an example of a driving simulator.

Fig.1.1 A driving simulator (Carlton)

The definition of a real-time simulator is not an easy one. There are quite many definitions of what makes a system a real-time system, but a more clear definition is that of between a soft and a hard real-time system. In a hard real-time system, one failure or missed timing will lead to a complete failure of the system. But in a soft real-time system, a single missed deadline or error will not cause a catastrophe. So the real-time environment built in this project can be classified as being a soft real-time simulator environment. (Laplante 2004)

Training simulators however do not necessarily have to be real-time simulators at all. For example in a simulator where the user practices making diagnostic decisions, there is maybe no need for a real-time simulation as it does not matter if

the response to a decision comes in 1 second or 3 seconds. On the other hand a training simulator for landing a shuttle manually will need some form of real-time behaviour to be realistic. Simulators can be used also for many other purposes besides training, for example system design and prototype testing.

Simulators can be more traditional devices with the user not immersed into another world, sitting for example on a car seat using real controls and only seeing an artificial view to the outside of the car. Virtual reality techniques and devices can be used to "sink" the user deeper into the artificial environment, so sometimes simulators can actually be called virtual reality environments, as the definition seems to be mostly dependent on the level of immersion. Between the full reality of the real world and the virtual reality there is also augmented reality, where the virtual non-existent objects of graphics, sounds etc. are added on top of the users perception of the real world, so that the user is in the middle of both real and the virtual world. Following are descriptions of the main components used in real-time simulator environments.

A motion platform creates the feeling of movements and accelerations for the user in the simulator environment. There are many different types of platforms, one of the most common being a Stewart parallel robot used as the platform. Shown in the Fig.1.2 below is a Stewart platform.

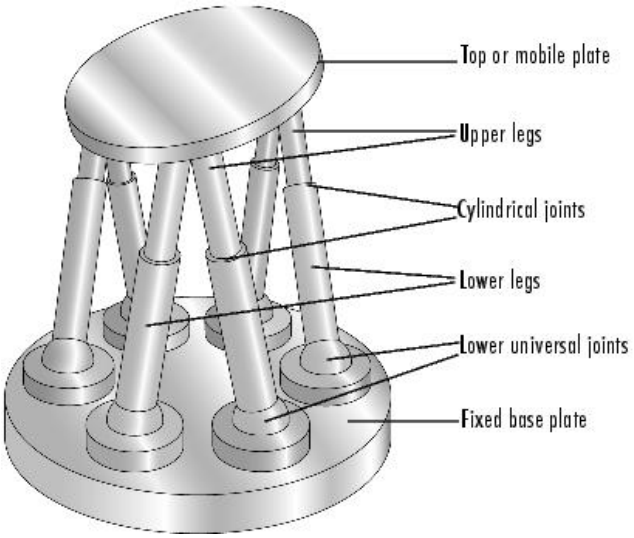

Fig.1.2 Stewart motion platform (Mathworks 2008)

The fixed base plate is usually attached to the floor or moving on some other moving platform or sometimes having even wheels underneath. The top plate is the part where the user of the simulator environment will be placed and it will create the feelings of motion and accelerations as well as different tilt angles. By adjusting the lengths of the actuators between the base and top plate, the position and the angle of the top plate can be altered in all six degrees of freedom. These platforms are in use from very small devices for one person or one part of body to devices that have a big vehicle or an airplane on top of the top plate.

In addition to movement, one of the most important methods of creating the feeling of reality is vision and thus it is also one of the most developed and researched parts of simulator systems.

Visual world of the simulators is created for example with either displays like normal monitors or groups of them, projected displays or by a display mounted on the head of the user. Below is a picture of a head mounted display.

Fig.1.3 A head mounted display (i-O Display Systems 2009)

Projected or flat screen (for example LCD) displays can be located either on the moving platform and thus the display is moving too or in front of the moving platform and staying still while the platform is moving. Displays attached to the users head are for example either small displays located in front of the eyes or devices that project the display into the user’s retina.

In Fig.1.4 is the projected display used in this project.

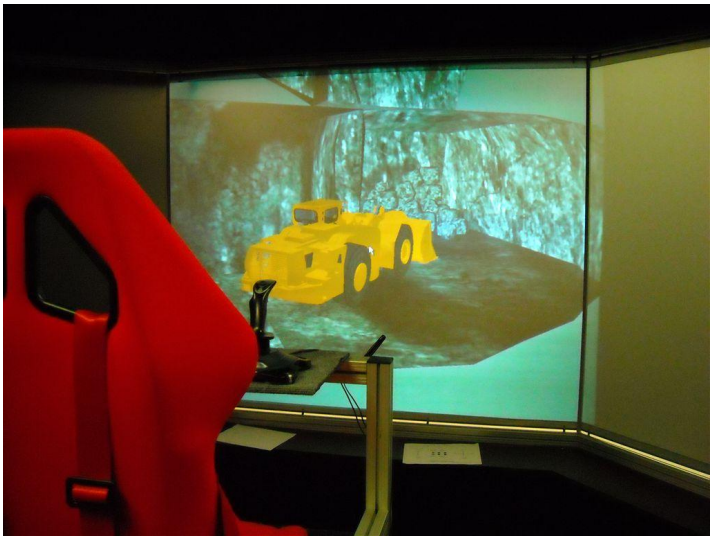

Fig.1.4 Projected display used in this project

In a simulator or virtual reality environment it is sometimes required to know the position or the posture of the user. When using a head mounted display, the position of the users hand is needed to show the hand in the graphics displayed, as the user cannot see his hand. Position, orientation and posture of the user is usually provided into the system with the help of global positioning system (GPS) or other similar positioning systems and with smaller scale devices like ultrasound position sensing, gyroscope with accelerometer or a haptic device.

In addition to touch and vision the third main sense for human is hearing. Thanks to movie industry and home theatres development, with current technology 3-dimensional sound environment can be realised without big problems. Affect of sounds on humans have also gathered a lot of interest and also plenty of research has been made. For machine design, the sound environment can bring new aspects as it has been noted that in virtual and augmented reality as well as in more traditional simulators, with good audio environment design it is possible to enhance the human performance. (Zhou 2007) Decent 3D environment can now be achieved even with relatively inexpensive equipment meant for home movie theatre use.

In order to be able to interact with the simulator or virtual reality environment, there needs to be some physical connection between the user and the environment. For more traditional simulators they can be physical replications of the actual devices controls. In more advanced virtual/augmented reality type simulators for example haptic devices, exoskeletons and glove based systems are used. Below is Fig.1.5 of an exoskeleton hand made in Germany.

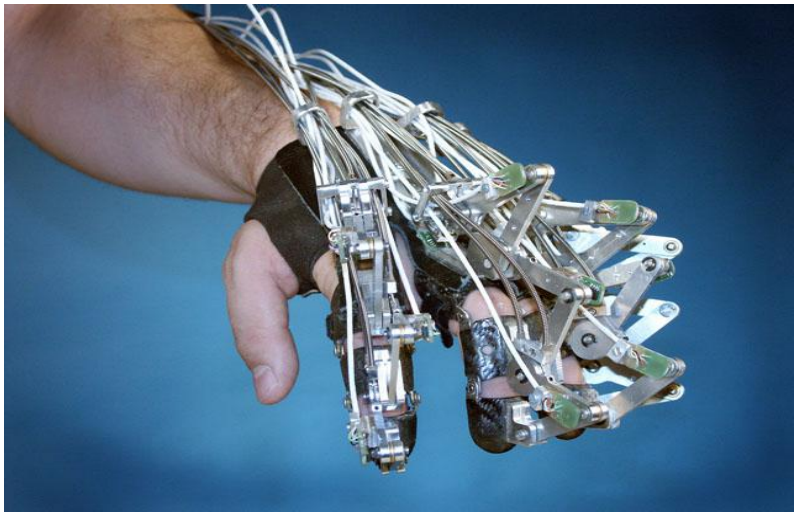

Fig.1.5 Exoskeleton hand (Wege 2009)

The choice of what kind of equipment is used depends on the actual system simulated, the reality level desired, the level of immersion needed and of the cost of the system.

1.2 Initial Situation

This work was made as a part of the Kvalive project. The participants in this project were Sandvik, Lappeenranta University of Technology (LUT), Mevea Oy, and VTT. LUT had acquired from different sources all of the components needed to build a real-time simulator environment and demonstrate it with a simulation of a Sandvik loader LH410 in a small mine.

In Fig.1.6 is the simulator's 3D model of the loader.

Fig.1.6 Sandvik loader 3D model by VTT

Most of these components had been tested separately to be functional and some were still waiting for someone to do so. The motion platform was already built at LUT, but originally it was not fully functional and Professor Huapeng Wu from LUT was assigned to make it functional for this project. MSc. Joni Sallinen had also earlier done work on the software components related to the Mevea's part in the project.

1.3 Target of the project

Target of the work was to integrate all the components that LUT had obtained, into a functional real-time simulator environment with a demonstration simulation of the Sandvik loader.

1.4 Work plans

1) Test the functionality and safety of the motion platform

2) Drive the motion platform with inverse kinematics from Matlab / Simulink using dSpace embedded computer

3) Install all software components into one computer and test that they work together while running them at the same time in the same computer.

4) Design and implement the needed programs to connect the motion platform and the haptic device to the main computer

5) Modify the 3D Virtools models made by VTT to be used with the Stewart motion platform, Phantom haptic device and 2 Logitech joysticks

6) Test the system with users to find out needs for improvement and what would be required to make the system as realistic as possible

2. DESCRIPTION OF THE REAL-TIME SIMULATOR ENVIRONMENT

2.1 General overview

The real-time simulator environment at Lappeenranta University of Technology was made of a motion platform, a main pc computer, motion platform user interface computer, haptic manipulator device, audio surround system, three joysticks, 3 projectors with rear projection screens and an embedded dSpace computer.

Software part of the environment was composed of the Virtools software, dynamics solver of the Sandvik loader, Matlab/Simulink program, Control Desk for creating a user interface for the Simulink model, a few additional building blocks for Virtools and 2 server programs to connect the Stewart platform and haptic device to the PC. Next the purpose, relations and functionality of all the components will be explained in detail.

2.2 Components of the system

In Fig.2.1 are the main components of the real-time simulator environment and their interconnections. Most of the programs run on a single PC which acts as a mediator of data between different programs and devices.

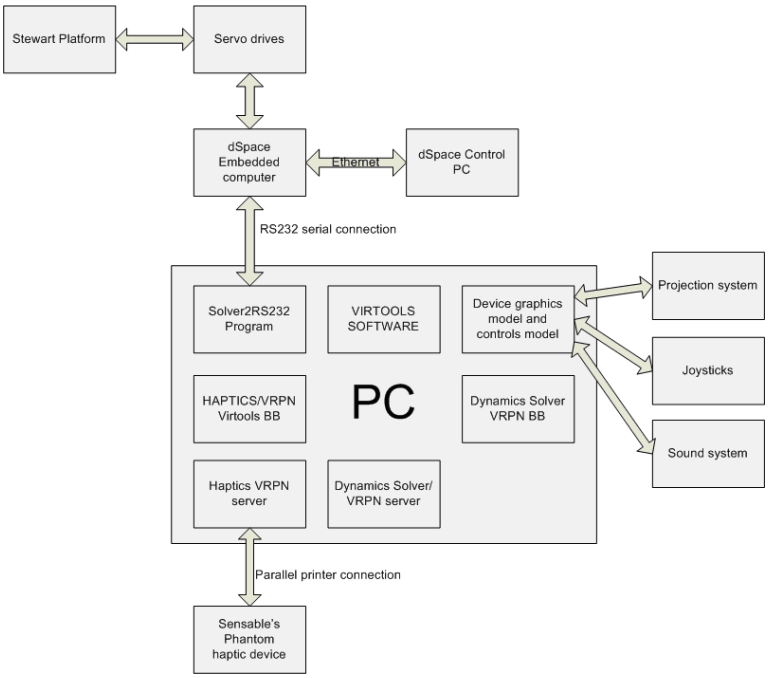

Fig.2.1 Main components of the real-time simulator environment

2.2.1 Hardware

A 6 DOF Stewart platform with six ServoMech UBA2RN1 ball crew linear actuators and AC servo motors with 200 mm movement range was used as the motion platform. The motors were equipped with incremental position sensors. In Fig.2.2 and Fig.2.3 are the motion platform and the actuators.

Fig.2.2 Photograph of the motion platform

A rally car seat was installed on the platform with 2 joysticks and 4-point seatbelts. Seatbelts were decided to be mandatory as it was noticed that the platform could do such high accelerations, that the safety of the driver had to be guaranteed somehow. Also a protection for the feet of the driver was made from plywood to make sure that driver's feet do not touch the actuators at any circumstances.

The linear actuators in Fig.2.3 had two sensors each to detect the end positions of the actuators. Since there was no absolute position of the actuator due to the use of incremental encoders, there was a need to get some information of actuators reaching their end positions.

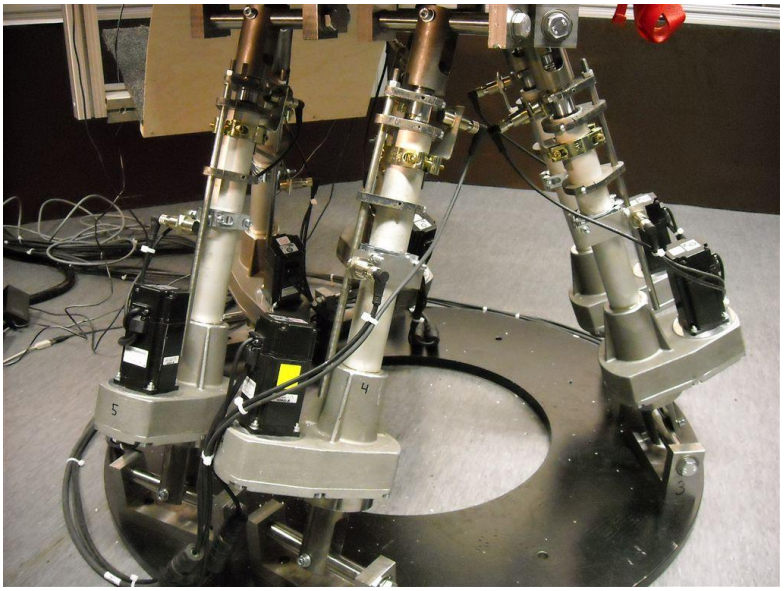

Fig.2.3 Motion platform actuators

The incremental position sensors and the motors were connected to six OMRON SGDH-04AW-0Y Servopack servodrives. The drive modules take care of the main control loop of the individual motors and of the security functions of the platform. The linear actuators limit sensors mentioned earlier, an emergency switch and a digital signal input from the controlling computer were connected to the drive modules.

If the limit switches were triggered by actuators movement to endpoints, the servo modules would stop the movement and allow movement only to the other direction. The emergency switch stops all movements of the platform and shuts down the motors immediately. The digital signal from the controlling computer was used to enable or disable the motors. As default when powering up the system, the motors were disabled according to this signal.

In Fig.2.4 is the cabinet with the power electronics and servomodules for the motion platform.

Fig.2.4 Cabinet for servo modules and power electronics

The drive modules were connected with six analogue +- 10 V signals for control and 6 square wave signals for position information for the dSpace embedded computer (Fig.2.5). The dSpace computer was running the software made with Simulink for controlling the platform, on a real-time operating system. This computer was connected to a programming and monitoring PC with Ethernet connection and to the main computer of the environment via RS232 serial connection.

The main PC was a computer with a quad core CPU and powerful graphics processing features to be able to run the programs needed for the real-time simulator environment. Like with modern 3D-games, the more power in total, the more realistic gaming experience you can get, seems to be true also in this case.

The main PC could display graphics either with one or 3 monitors or 1 to 3 projectors. There were 3 additional I/O-cards installed to the computer, one pulse counter card, one analogue output card and one analogue/digital I/O card. The pulse counter card was used for reading the position pulse information from the motion platform actuators incremental encoders. The analogue output card was used for sending the reference values for the servo drives from the main PC and the analogue/digital I/O card was used for providing the enable/disable signal for the servo modules. In the final solution of the project, these were however not used in the system. Computer was attached to the dSpace computer via RS232 serial connection.

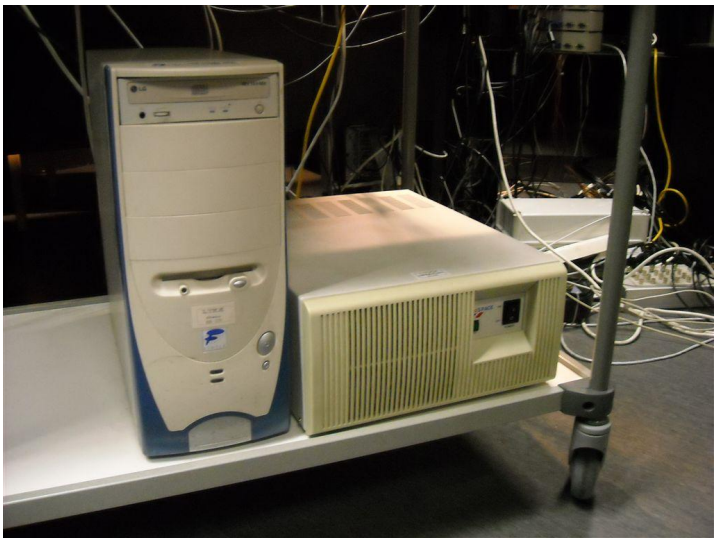

Fig.2.5 User interface computer (left) and dSpace computer (right)

Three Logitech's Force feedback joysticks were attached to the computer with USB connections. One joystick was to be used with the haptic device on the table near the main computer and 2 joysticks were placed on the table of the motion platform. The two joysticks in the Fig.2.6 were used like the two joysticks on the Sandvik loader. As the real loader had two special types of joysticks designed for the application, the full features of the real joysticks could not be fully implemented with these two.

Fig.2.6 Two joysticks to control the loader

Audio environment of the simulator was implemented with a Terratec external USB soundcard with Dolby surround 5.1 and an optical output signal. It was connected to a surround sound decoder and amplifier. Amplifier was connected to several speakers and one subwoofer behind the 3 rear projection screens of the simulator. With this type of sound system and Virtools it should be possible to obtain a relatively realistic spatial sound environment. In Fig.2.7 are the speakers behind the screen.

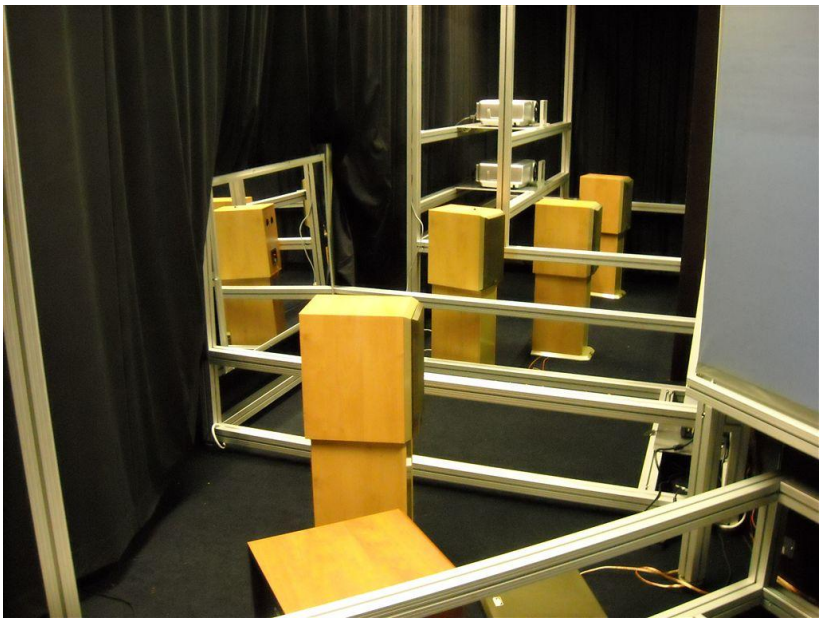

Fig.2.7 Speakers and subwoofers behind the rear projection screens

Below in Fig.2.8 are the three rear projection screens of the simulator.

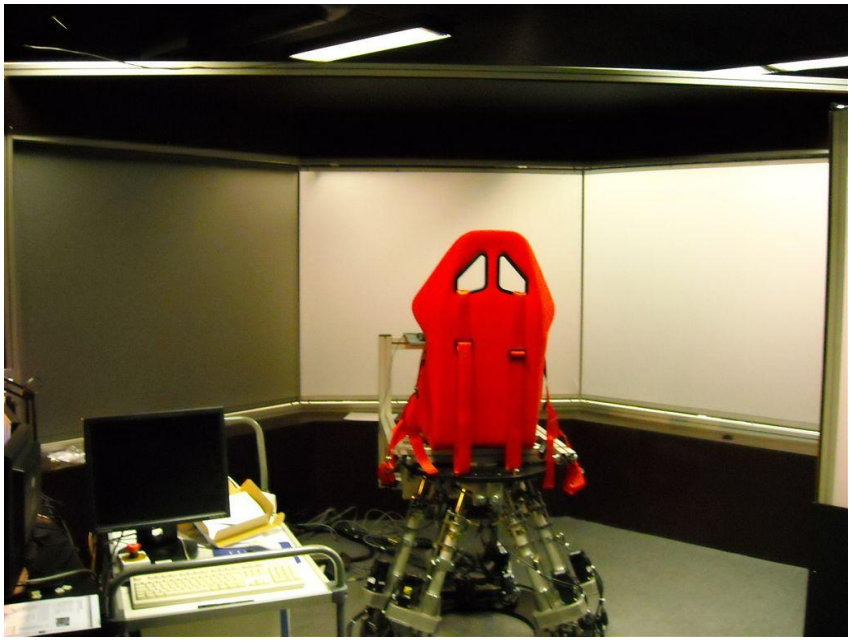

Fig.2.8 Three rear projection screens

The second important component of interest of this environment in addition to the motion platform was the Phantom Premium 1.5 HF haptic device by Sensable Technologies. It was purchased to LUT in a separate project. (Pellinen, 2008)

Haptics as a science could be called a science of touch or touching. The sense of touch has an important meaning for us, it is needed for understanding and exploring the real world. It has a significant historical meaning and also a meaning for the survival of our species. During human history, touch has been a welcome thing and sometimes it has been banned and been scared of.

Feeling of touch is sensed by a vast amount of sensors in our skin and in our muscles and joints. Together these provide us the sense of touch, position and movement of our bodies. Everyone's experience of the sense of touch is different and it also changes due to age, level of tiredness, due to illness etc. Visual and auditory sensations are easy to produce as they are gathered by single organs like eyes and ears, but the sense of touch is produced everywhere in our bodies. Sense of touch requires forces imposed on the human sensors to create the sensation.

It is more difficult to create then audio and visual sensations and is still not so often used method of interfacing to us humans.

A haptic device creates sensations of touch and forces to our muscles and skin. It creates a sense of touch, rigidity, weight, acceleration, forces and position of our limbs to our sensory system and brains. With haptic device it is possible to create the sensations that are received by our brains in our everyday lives while exploring our surroundings. But it is also possible to create sensations that have no base in the real world, meaning sensations that are completely artificial. This allows many possibilities of entirely new human machine interfaces. In simulator at least two types of sensations should be provided to be able to have a good level of reality. Every sense added to the real-time simulator increases the level of immersion, meaning the level of how deep the user “sinks” into the virtual environment of the simulator.

When creating the simulated environment in real-time, in addition to the visual rendering of the simulators 3D environment, the haptic environment needs rendering too. In haptic rendering are calculated the forces applied by the haptic interface against the user, depending on the positions and movements of both.

Haptic devices can be constructed in quite many ways. Some can be very specific devices that simulate only one type of haptic sensation. They can also be very universal devices that can provide the haptic feedback from many different things. These universal devices include for example VR gloves with force feedback, exoskeletons and devices with serial or kinematic actuators. The Phantom haptic device that was used in this project is a device that resembles a serial kinematic robot in construction.

In Fig.2.9 is a picture of a parallel kinematics haptic device.

Fig.2.9 Parallel kinematics haptic device (Novint 2009)

The Phantom Premium 1.5 HF haptic device by SensAble Technologies, Inc. was connected to the main PC with a parallel cable utilising the traditional data transfer method used with printers. The HF in the name tells that this device is capable of creating bigger forces then the basic models. The device can use 6 DOF with force feedback for 3 DOF. The user operates the haptic device by taking a hold of the stylus pen-like stick and moving it in any direction and changing its angle. When moving the stylus, the device can give feedback by creating forces in all directions to oppose the movement made by the user.

In Fig.2.10 the movement and rotation possibilities of the haptic device can be seen.

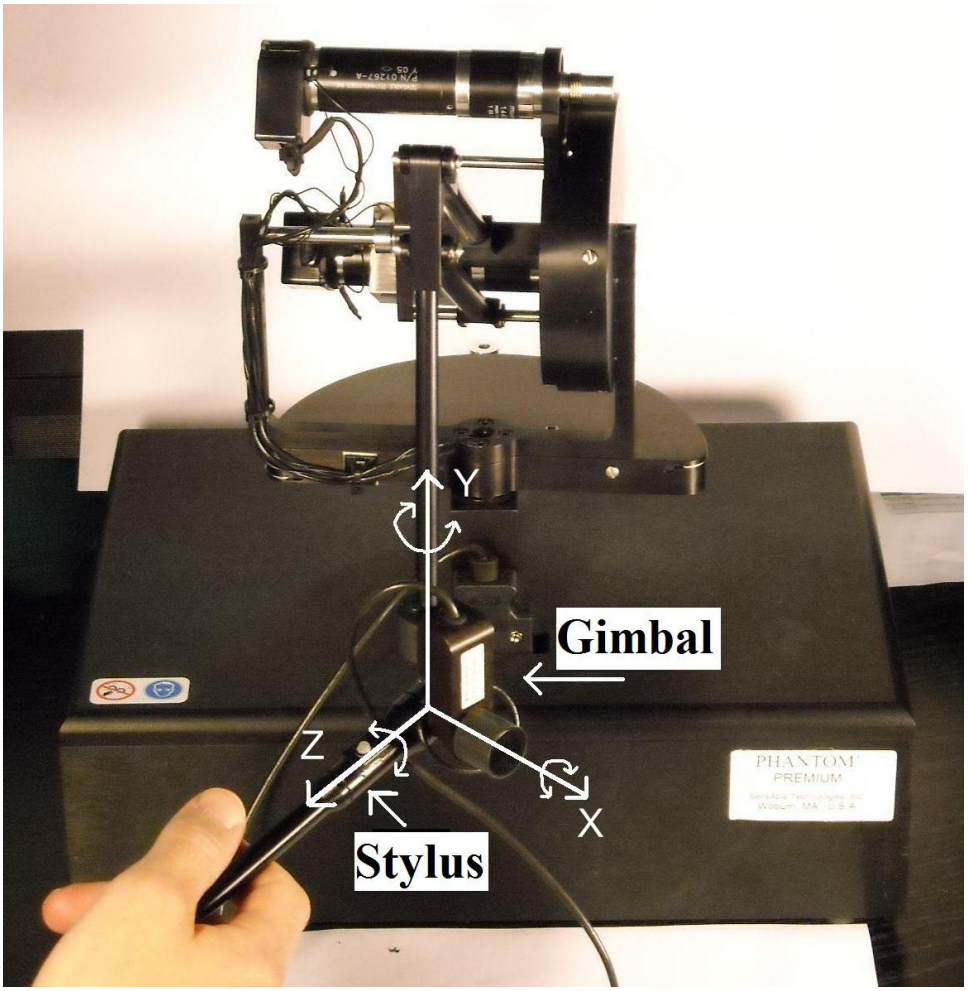

Fig.2.10 Motions of the Phantom haptic device

Main coordinates xyz are in millimetres and their equivalent angles in radians. They are given by the device for the middle point of gimbal, representing the endpoint of the stylus. Also velocities of the xyz axles in mm/s and angles of the main joints in radians can be retrieved from the device.

2.2.2 Software

Fig.2.11 shows all the software components and their interconnections in the simulator environment. The Sandvik loader's graphics and behavioural model, the user controls and both haptics and dynamics solver's building blocks (BB) are run inside the Virtools software.

The server program for haptic device, the dynamics solver and Solver2RS232 program are all separately running programs on the same computer as the Virtools. All of these could also be run from another computer. In the following paragraphs these separate programs, Virtools building blocks and also the program running in the dSpace computer will be explained in more detail. Virtools and models will be explained in the next chapter.

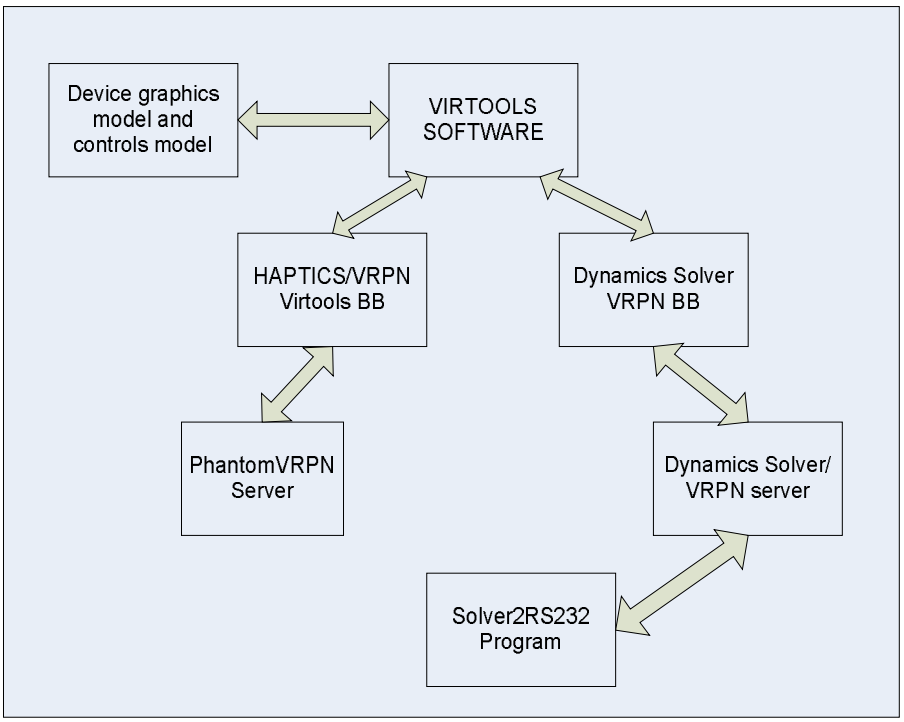

Fig.2.11 Software components in the systems

VRPN (Virtual reality peripheral network) is an open source IP based protocol designed to combine and connect a wide variety of software and hardware components together to form a functional and modular virtual reality environment. Connections are made externally with standard Ethernet or with wireless network connections and inside a single computer they are made as normal socket connections. All components are either servers or clients or both. There are several ready defined classes of devices that one can adapt their device or program into. Making components this way is modular and the components are easier to connect to each other. VRPN's creator University of North Carolina at Chapel Hill, provides for free c++ libraries that include the main functionalities of the VRPN connections

for many different platforms like Windows, Linux and some RTOSs making the implementation even easier for new devices.

Dynamics Solver/VRPN server is a separately running program made by Mevea Oy. The dynamics solver is a very essential part of a successful simulator as it models the dynamic features and behaviour of the simulated machine and in our case it provides the required position and orientation information to both the Virtools program and the motion platform. For connecting to other programs it uses the above mentioned VRPN as either a server or a client. This component was included as a so called black box, only the interface of the VRPN connection was known. In Fig.2.12 is the user interface of the dynamics solver. From the window one can start, stop and pause the solver, monitor different variables of the simulated machine and monitor the VRPN connection status and events.

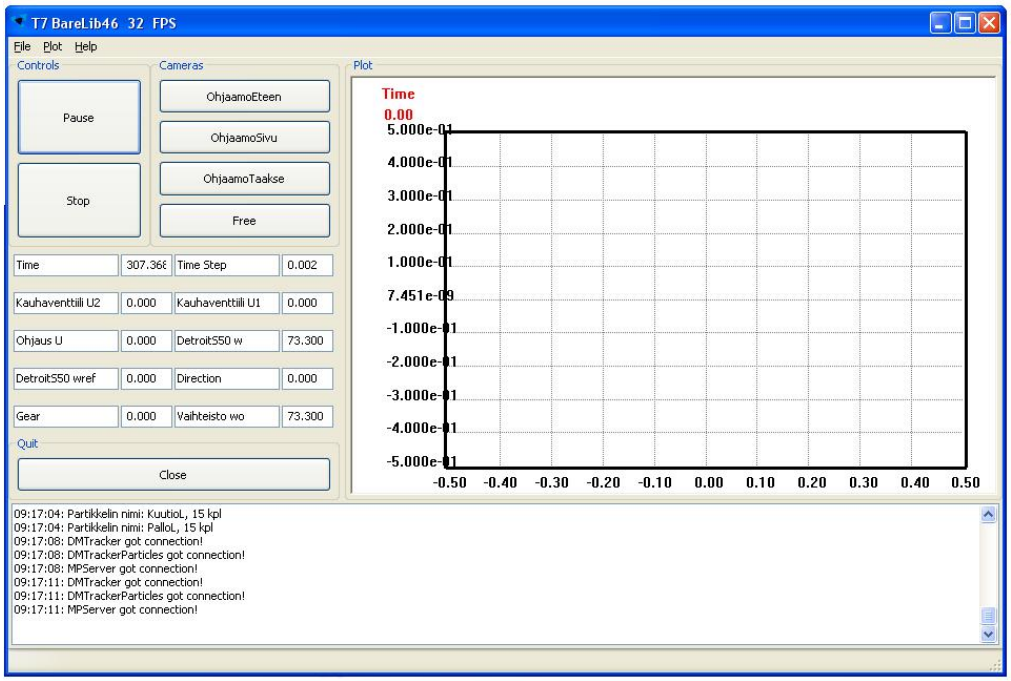

Fig.2.12 Dynamics solver by Mevea Oy

Dynamics Solver VRPN building blocks were made for the project by MSc. Joni Sallinen to be used as connecting building blocks between the Virtools model and the dynamics solver program described earlier. These components utilised the freely

available VRPN libraries to implement the VRPN functionality. In Fig.2.13 these components are shown in the Virtools model.

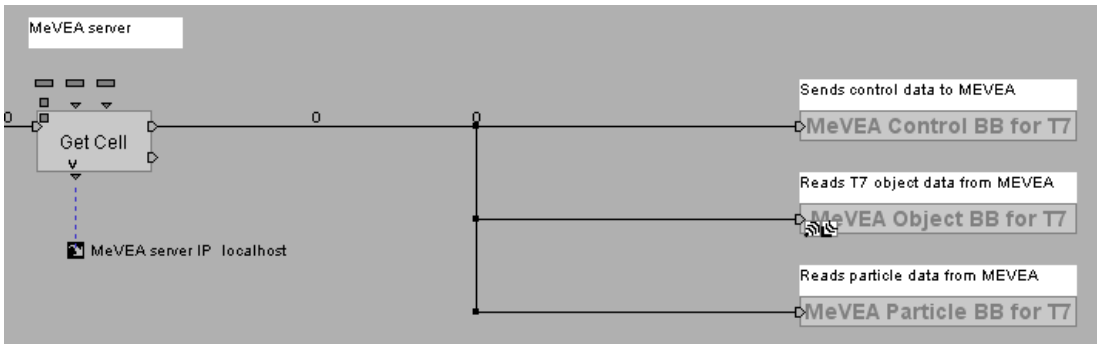

Fig.2.13 Dynamic solver building blocks in the Virtools model

In Fig.2.14 is a flowchart of the main threads of the PhantomVRPNServer program made as a part of this thesis. This program is a standalone program to connect the Phantom Premium 1.5 HF haptic device to the virtools model using a VRPN connection.

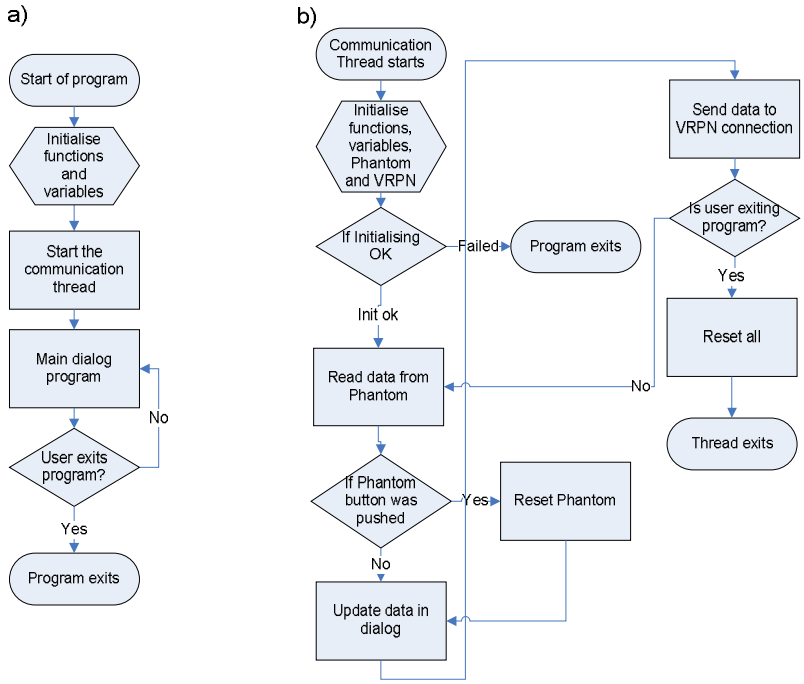

Fig.2.14 Flowchart of the main threads of the PhantomVRPNServer program

Haptics part of the program was realised using libraries provided by the Phantom's manufacturer Sensable Technologies Ltd. and VRPN connection was done by using

the freely available VRPN libraries. This program reads the coordinate, angle and button information of the Phantom haptic device and sends it to the receiving VRPN device or program. Functionality of the threads is explained after the picture. In part a) of the Fig.2.14 is the main dialog thread of the program. After starting the program, all program window variables and functions are initialised along with the communication thread. Next, the main program starts the communication thread described in part b). Then main program checks if the user wants to end the program and cleans it up when exiting. After starting the communication thread, first the Phantom haptic device and VRPN connections are initialised. If the initialisation fails, the program exits. If all initialisations went normally, thread enters the main loop. In the loop, the thread reads the current values of coordinates, angles and button state from the haptic device. Second, it checks if the haptic device's button was pushed and if so, it resets the coordinates and angles of the haptic device to zero values. Then the data is updated to the programs dialog window and after this it is sent to VRPN connection and to possible VRPN clients requesting the information. If user is exiting the program, this loop is left and the haptic devices' and VRPN's connections are closed and the thread is stopped and deleted. If the program continues to run normally, program loop again starts to run by reading the data from the haptic device. In Fig.2.15 is the user interface of the program.

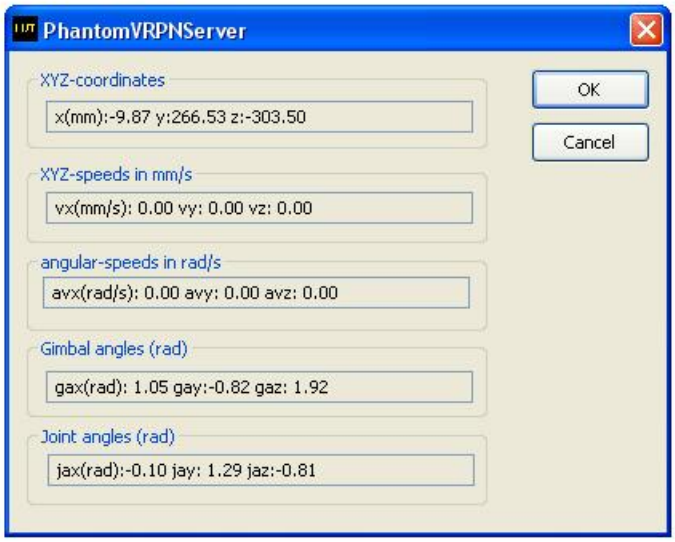

Fig.2.15 User interface of the PhantomVRPNServer

HAPTICS VRPN/Virtools BB is a building block for Virtools made as a part of this thesis to connect the previously described PhantomVRPNServer to Virtools with the VRPN interface. In Fig.2.16 are the two main parts of the building block program represented as flowcharts.

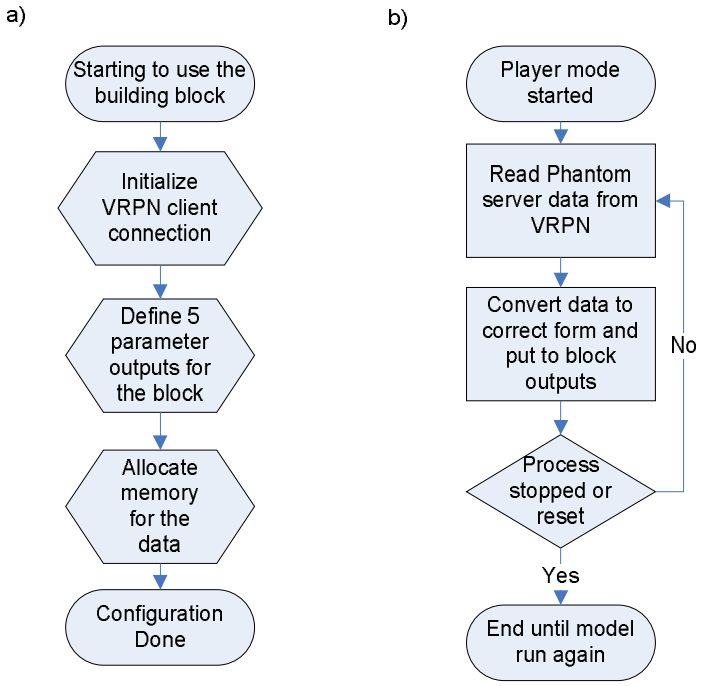

Fig.2.16 The two main parts of the HAPTICS VRPN/Virtools building block

Flowchart a) in the Fig.2.16 shows what happens when the building block is loaded into the model. First it initialises the VRPN connection, then defines the output and input connections for the block and allocates memory for the data received from the VRPN connection.

In flowchart b) in Fig.2.16 when the model is started to play, values received from the PhantomVRPNServer are received, then converted to a form suitable for the outputs of the block and set into the outputs. If process is stopped or reset, this loop ends and continues until play is started next time. Else this loop is repeated each time the Virtools calculates graphics, behaviour etc. for the model.

Below is a Fig.2.17 of the building block and its outputs as an example. The output values are all vectors with three values per each vector. The line coming from the left of the picture is a connection used to activate this block and the triangle connection on the right side of the block is equivalent output for activation of other blocks. Position XYZ contains the global xyz-coordinate values of the gimbal in millimetres and the velocity XYZ gives the equivalent velocities in millimetres/second. Both the gimbal and main joint angles are given in radians. Button state is either value 0 for not pressed and 1 for button pressed.

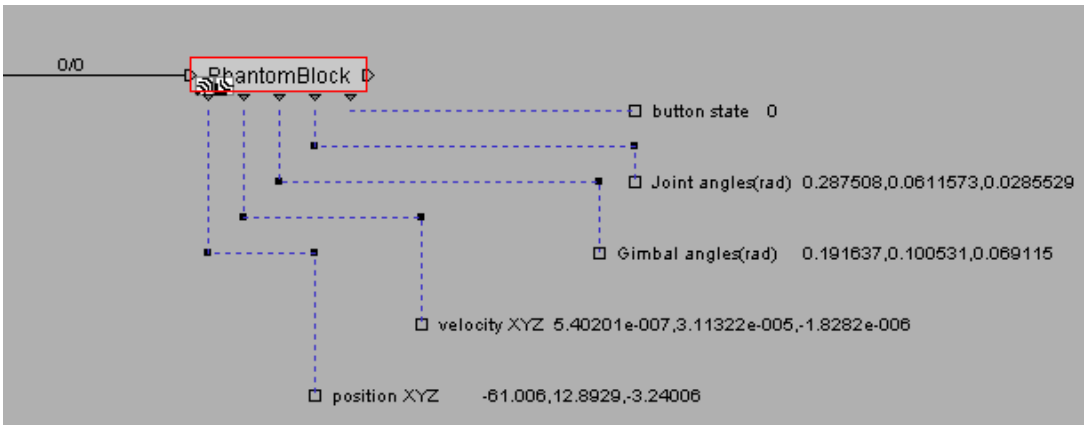

Fig.2.17 HAPTICS VRPN/Virtools building block and its outputs

The Solver2RS232 program is a separately running program that connects to the dynamics solver by Mevea Oy using a VRPN connection. It sends the solver data using a RS232 serial connection to the dSpace embedded computer which controls the motion platform. The program acts as a VRPN client towards the solver and retrieves position and orientation information for the motion platform. The information retrieved is 6 DOF position information of a point of the loader underneath the driver's seat. Solver2RS232 program also encodes the data so that it can be sent across the RS232 connection as fast as possible.

The two main threads of the program are shown in Fig.2.18. The main thread in flowchart a) starts with Windows' own dialog based programs initialisation functions and also initialises the thread responsible for the communication task.

After starting the thread in flowchart b), the main program thread just listens to if the user presses any button to exit the program and then cleans up when exiting.
The second thread in flowchart b) Fig.2.18 initialises and opens the RS232 serial connection and then the VRPN connection.

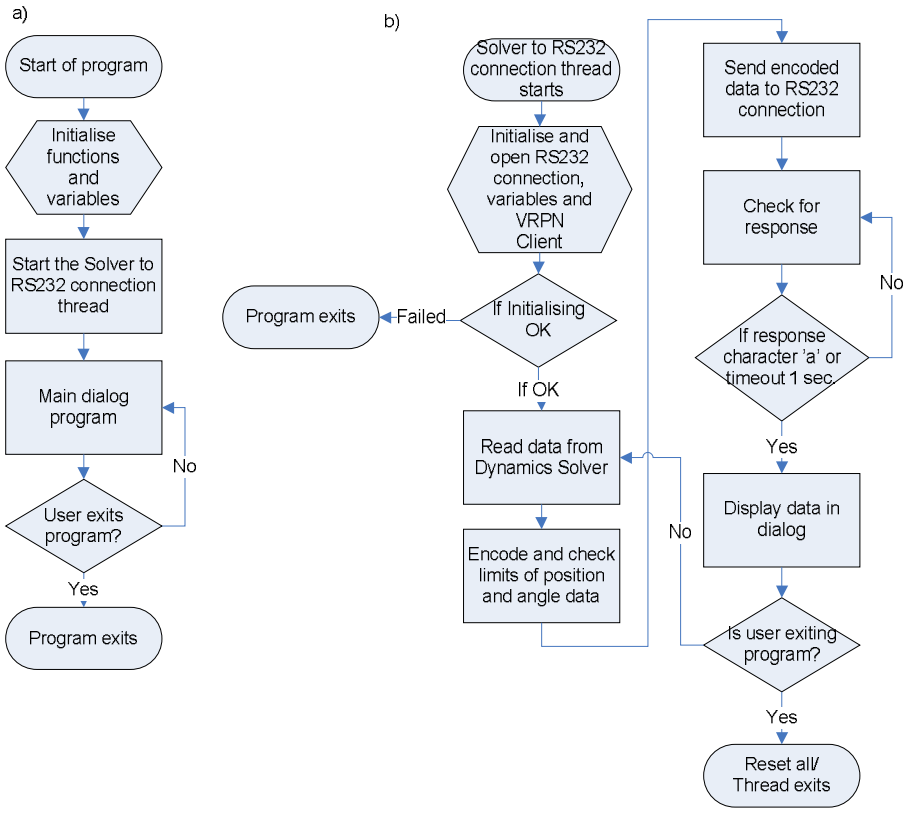

Fig.2.18 Solver2RS232 programs main threads

If initialisations were successful program continues execution. Otherwise the program gives user an error message and exits. Next the program asks for the data consisting of position coordinates x, y, and z and their respective angles alpha, beta and gamma from the dynamics solver program. After receiving the data, the program encodes the data into smaller amount of space and checks that the values are within accepted limits of +- 255mm and +- 25.5 degrees. The data is then sent via the RS232 serial connection with a speed of 115200 bits per second to the dSpace embedded computer.
When done with sending the data, the program begins to wait for a response from the dSpace computer to inform that data has been received and handled properly. The message to be waited is a character 'a'. If no character is received in 1 second, program will continue normally and if character 'a' is received earlier then 1 second,

the program will continue immediately. This way the transmission speed between computers can be adapted to the current processing situation of the dSpace computer as it has also the task of running the main controller of the Stewart platform. Maximum data package rate of transmission due to program limits is 1000 times per second. After transmitting the data, it is sent to the dialog thread to display to the user. And if the user is not exiting the program, the program loop will continue again from asking the data from the dynamic solver and receiving it. If program is closed, all threads are ended and all connections are terminated.

In Fig.2.19 is a flowchart that describes the encoding function of the Solver2RS232 program and in Fig.2.20 is the dialog of the program.

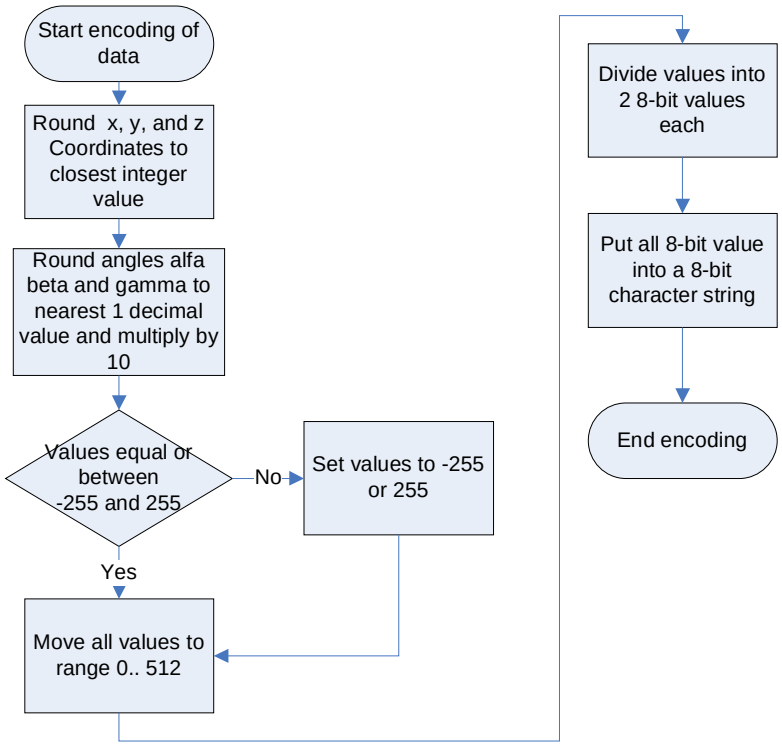

Fig.2.19 Solver2RS232 encoding of the serial data

Encoding is done between receiving the data from the dynamic solver and sending it to dSpace computer thru the serial connection. First the x, y and z coordinates are rounded to the nearest integer (i.e. 123.41 to 123) values and angles alpha, beta and gamma are rounded to the nearest 1 decimal accuracy (i.e. 15.23 → 15.2). Next the angles are multiplied by 10 making them integers and the values are checked that

they are within accepted range. Coordinates must be between -255 and 255 millimetres and angles must be between -25.5 and 25.5 degrees. If the values are outside this range they will be set to the maximum values. Then values are added by 255 to move them to range from 0 to 512. After this values are divided into two 8-bit bytes each with the following method. If value is bigger then 255 then the first byte is 255 in value and the next byte is the amount that the value is bigger then 255. For example 300 would be 255 and 45. If a value is equal or smaller then 255 then the other byte is 0 and the second byte is the value itself i.e. 240 → 0 and 240. Next each byte is put into a character string of 14 characters. First a start character 's', then the two bytes of coordinate x then the two bytes of coordinate y, up to the angle gamma, after which there will be a character 'e' signalling the end of the data package.

Example of the data package is as follows: “sxxyyzzaabbcce”, where xxyyzz are coordinates and aabbcc are the angles. In total the amount of bytes is 14 which is the size of the receiving hardware buffer of the dSpace embedded computer, thus enabling the dSpace computer to receive all data with one read of its’ buffer. As the speed of transmission for one bit (baud rate) was 115200 bits per second and there was one start and stop bit, the maximum transfer rate of coordinate and orientation information can be calculated with the following equation 2.1.

$$Transfer\_rate = \frac{baudrate}{(bytes \times (bp + sb + eb))} \tag{2.1}$$

where, $bytes$ is the amount of bytes to transfer, bp is the amount of bits per byte (normally 8), sb is the amount of start bits, eb is the amount of end bits and $baudrate$ tells how many bits per second can be sent.

As a result of the calculation, the maximum transfer rate of the rs232 transmission line for the coordinate and orientation information is 822 times per second. However due to delays and undeterministic nature of Windows and PC hardware, this speed is not possible to obtain most of the time (Zhang 2005). If higher speed of transfer is needed, removing the feature of displaying the values to the user in dialog would speed up the program somewhat.

In Fig.2.20 is the user interface of the Solver2RS232 program. The values in the interface are updated at the same rate as the dSpace computer accepts data.

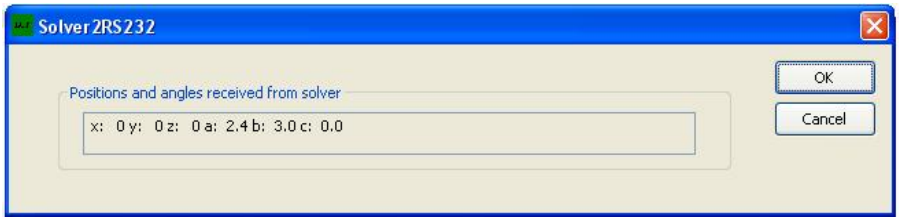

Fig.2.20 The user interface of the Solver2RS232 program

In the Fig.2.21 is the Simulink model of the controller for the motion platform. This model was downloaded to and run from the embedded dSpace computer.

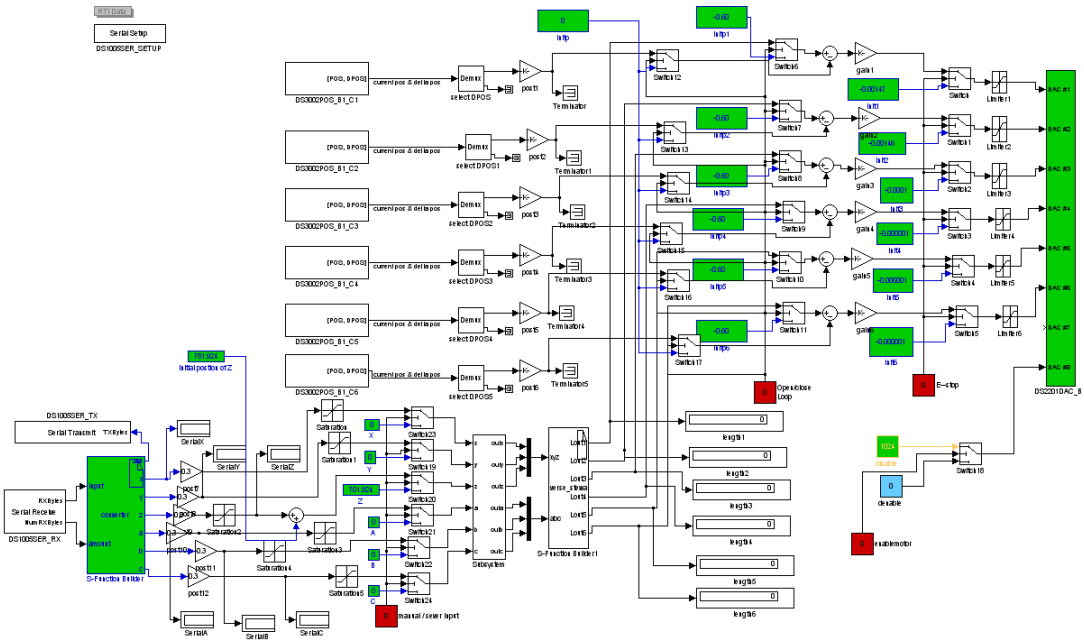

Fig.2.21 Simulink model of the controller of the motion platform

In the model, device specific blocks were used that enabled the use of dSpace computers hardware from a normal simulink controller model. After compiling the model it was downloaded thru ethernet to the embedded dSpace computer and run from there. The computer used for modelling and downloading the program was

then used for running a user interface made with the Controldesk program. This graphical user interface enabled the user to monitor and modify settings and variables in the model that is running inside the embedded computer.

In Fig.2.22 is shown the user interface made for the motion platform.

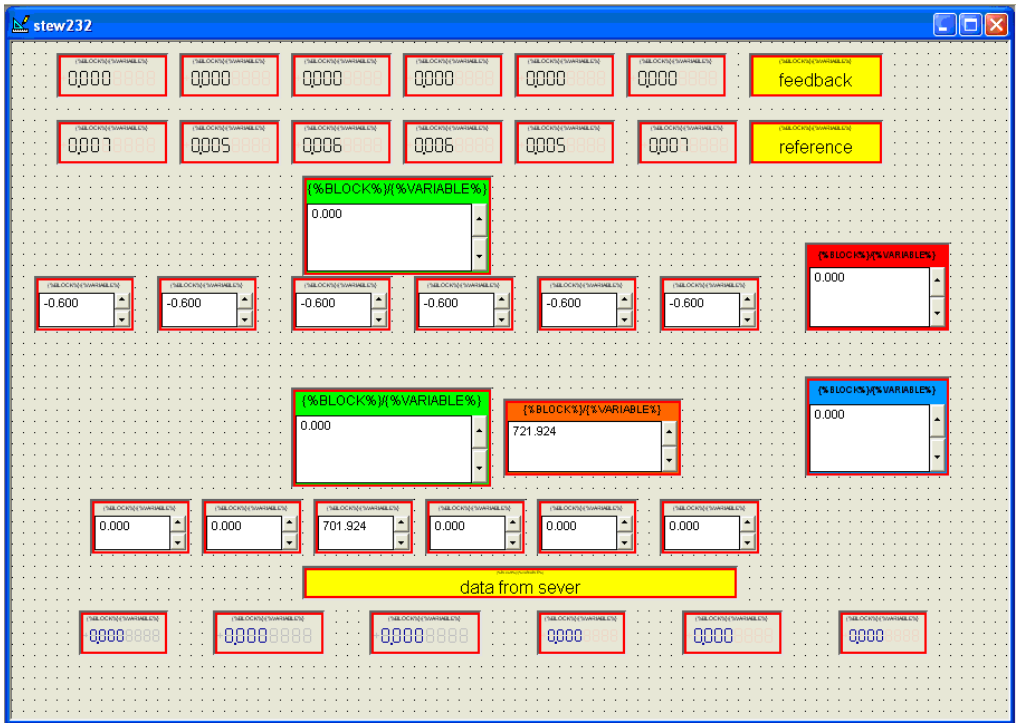

Fig.2.22 The user interface made for the motion platform

The user interface for controlling the dSpace computer and the motion platform was done with a Controldesk program that is included with the dSpace software package. In the program user loads a file provided by Simulink that tells the program what features and parameters are available from the model downloaded to the dSpace computer. Then user chooses from a library the needed interface components like buttons etc. After placing them into the window, the variables loaded from the file are connected to the user interface items. When this is done, the user interface is run just by pressing a button. When run, it connects to the dSpace computer via ethernet.

In Fig.2.23 is shown the flowchart of the Simulink model for controlling of the motion platform.

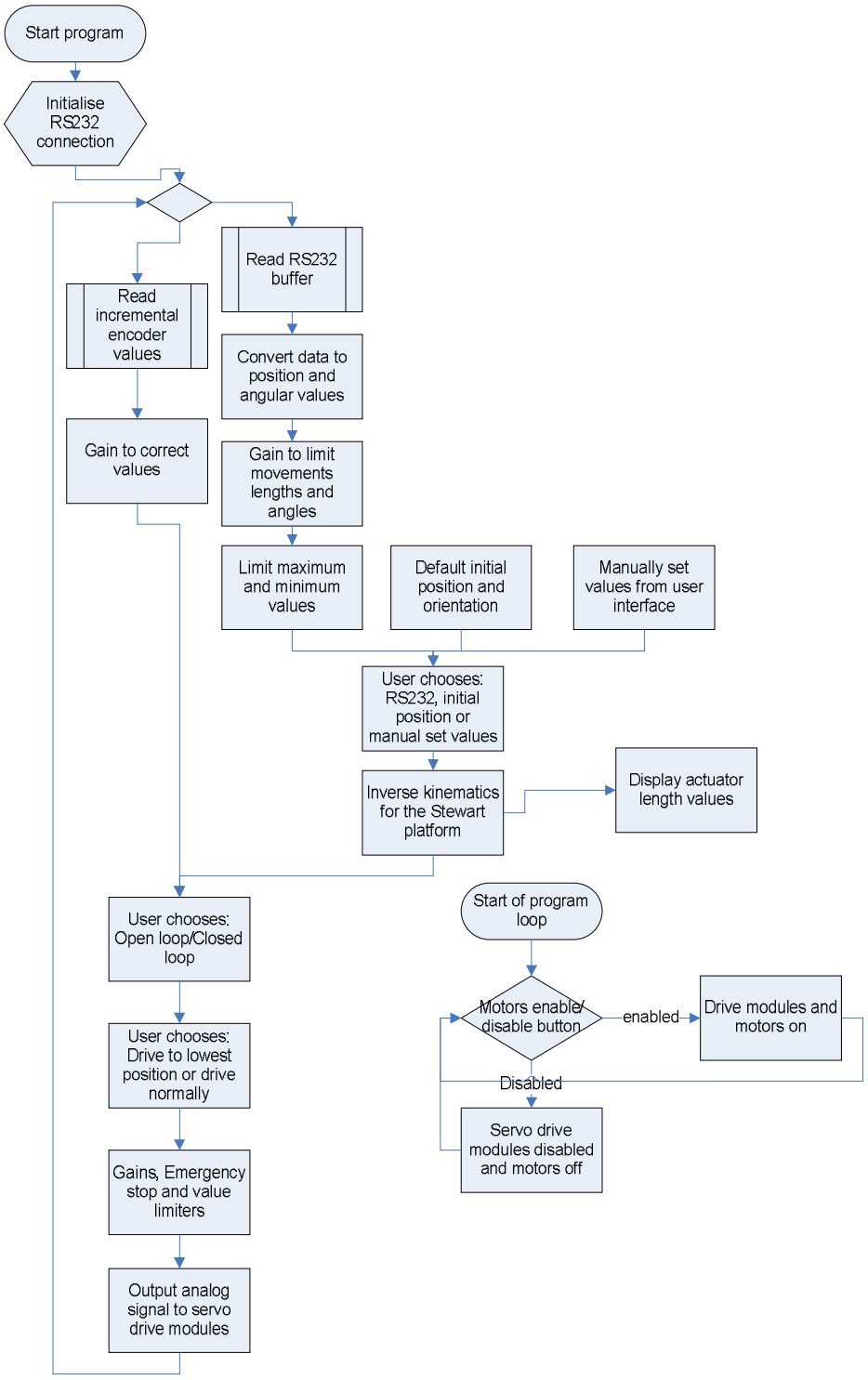

Fig.2.23 Flowchart of the Matlab/Simulink model for controlling of the motion platform

As the model starts to run, the hardware of the RS232 serial connection, analogue outputs and incremental encoder inputs are initialised. After initialisations, all analogue inputs are read and outputs set at the end of every execution cycle. Inside the execution cycle between the inputs and outputs, gain values are used in several

places to calibrate and limit different values. After reading the 14 byte buffer of the RS232 serial connection, the received data is decoded (described later) into the original position and orientation values. Movements are then limited by an experimental gain value and maximum and minimum values to make the platform movements feel more comfortable to the user. The reference value to the controller is taken next from either the received value from serial connection, from a default initial position which is located in the middle of the platform working space or from a manually set value depending on the user selection. The position coordinates and orientation angles are then converted into actuator lengths with inverse kinematics calculations.

Inverse kinematics was calculated using the following equations (2.2 and 2.3):

$$\vec{d}_{\mathrm{i}} = \vec{P} + R \bullet \vec{b}_i - \vec{a}_i \tag{2.2}$$

where: $\vec{d}_i$ is the vector containing actuator length components, $\vec{P}$ is the vector between base and moving platforms coordinate systems, R is the rotation matrix of size 3x3 for 6 degrees of freedom, $\vec{b}_i$ is the vector between joint i and frame origin of the moving platform, $\vec{a}_i$ is the vector between fixed base frame origin and joint i

From equation 2.2 the vectors describing each actuator can be calculated with the end effectors values of position and orientation as input. Equation 2.2 is changed into a form (equation 2.3) from which the actual lengths of the actuators can be calculated.

$$d_{\mathrm{i}} = \pm\sqrt{\left[\vec{P} + R \bullet \vec{b}_i - \vec{a}_i\right]^T \bullet \left[\vec{P} + R \bullet \vec{b}_i - \vec{a}_i\right]} \tag{2.3}$$

where d_i is the scalar value length of a single actuator i. (Wu 2008, Tsai 1999)

In Fig. 2.24 are the platform geometrics, showing all vectors and coordinate frame origins.

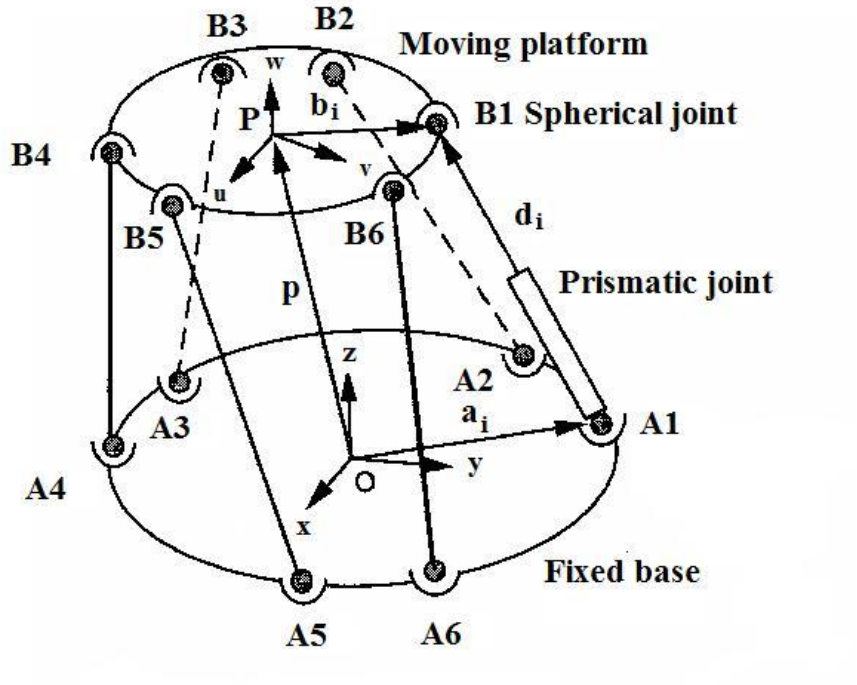

Fig.2.24 Stewart platform geometrics (Tsai 1999)

Desired position and orientation of the middle position of the moving platform in relation to the global coordinate system are used as equation 2.3's input values. The resulting values describe the length values of the actuators that are required to drive the motion platform into the desired position and orientation.

Following flowchart (Fig.2.25) describes the calculation in the c-code of the simulink block for inverse kinematics. First, all the variables are created and initialised. Position coordinates of the joints of the moving platform and base are then input into matrices and angle values are converted from degrees to radians. With inverse kinematics calculations the vector components for each of the actuators are calculated and after this they are combined to give scalar length values of the actuators. The actuator lengths are then reduced by the initial lengths of the actuators to get the values needed to input as reference values to the controller of the motion platform. (Wu, 2008)

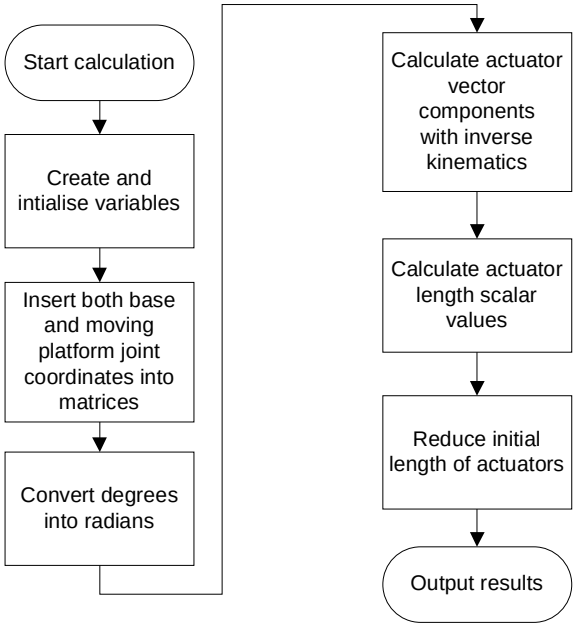

Fig.2.25 Flowchart of the inverse kinematics calculation for the Stewart platform

After calculation of the inverse kinematics, the actuator lengths are shown to the user in the graphical interface. User can choose between a closed and an open loop control, which is default to open loop so the platform can be initially operated safely despite wrong initial values possibly coming from the incremental sensors. When the controller is operated in the mode following the received reference values from the serial connection, the closed loop mode must be used. Also included into the controller are an emergency stop button and a switch with which the motors can be enabled or disabled. Initially the motors are disabled as a safety feature. As the last thing in the execution loop, the calculated output values are set into the analogue outputs of the dSpace computer, which are connected to the servo controller modules.

In (Fig.2.26) is a flowchart that describes the decoding of the data that arrives from the main computer running the dynamics solver to the embedded dSpace computer.

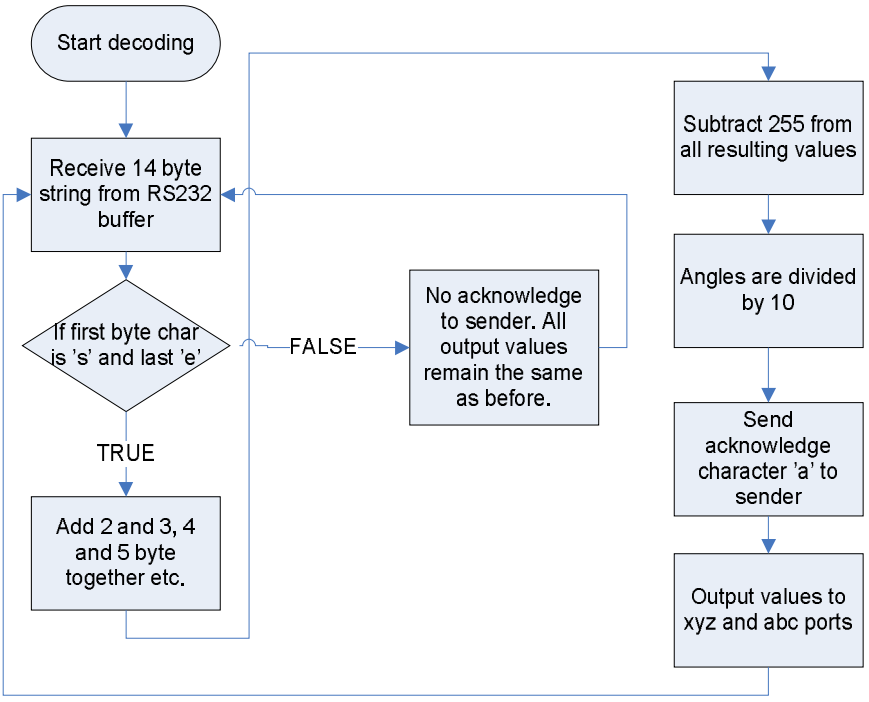

Fig.2.26 Decoding the RS232 serial data arriving to the motion platform controller

First the received 14 bytes are put into a character string of 14 bytes in length. The program checks if the first character is 's' and the last is 'e'. If not, the data is discarded, no acknowledge is sent to the sender and all output values are kept the same as previously. If characters 's' and 'e' were found, program continues the decoding normally. All bytes are added together in pairs forming 6 values in total. For example byte1+ byte2 results in coordinate value x, byte3+byte4 results in coordinate value y etc. After combining the bytes, 255 is subtracted from all the values to shift them from range 0 to 512 back to -255 to 255. Angle values are then divided by 10 to add the one decimal back to the values. The acknowledge character 'a' is sent to the sender and output values of coordinates x, y and z and angles alpha, beta and gamma are sent forward to be processed in the embedded computers execution loop.

2.2.3 Virtools

Virtools is a program by Dassault Systèmes which can be used to combine the many hardware and software components of any simulator / virtual reality environment together. It allows the user to combine 3D graphics made in other programs, their dynamics, behaviours and other aspects (sounds, controllers etc.) with a graphical programming interface. This programming method is based on placing graphical blocks in correct order and connecting them together with connecting lines in a way desired to realise the functionality wanted. In Fig.2.27 the block method of programming a model is shown.

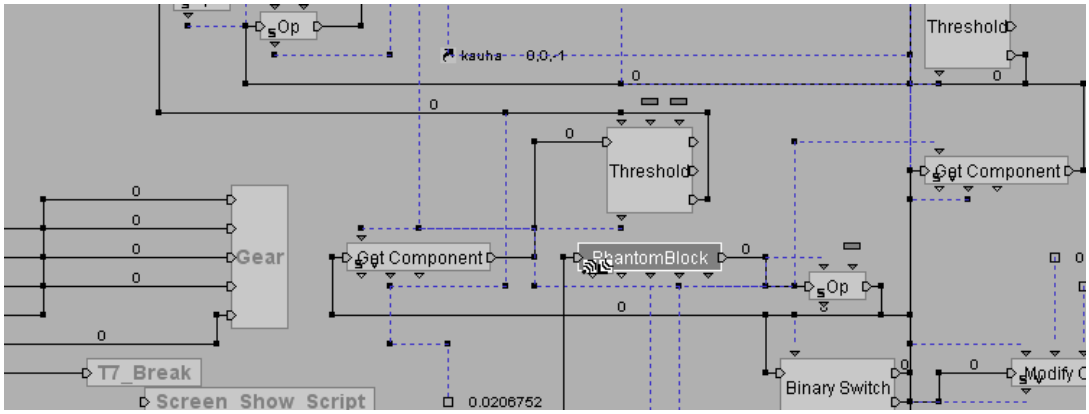

Fig.2.27 Example of Virtools graphical block programming

A library of blocks is already provided with the program to enable different kinds of behaviours/interactions of the 3d objects and also of the external devices connected to the system. Tools and blocks to make your own user interfaces for www applications and for virtools itself are also provided. Main programming methods are the already mentioned graphical block and connection programming, script programming with a simple scripting language and programming with c or c++ language. Programming new building blocks to the library is done with C++-language using Microsoft Visual Studio 2003 and Virtools provided project wizards for building the frame and project files of the blocks. The building blocks are implemented as windows dll's.

The model that contains the mobile machine and its' surrounding graphics, behaviours, sounds and user controls is the central part of the simulation of the Sandvik loader. The model used was originally made by VTT and as a part of this thesis it was modified at LUT for the purpose of using it as a demonstration of the real-time simulator environment at Lappeenranta University of Technology. The two versions of the modified model included all the 3d graphics of the loader and their properties and behaviours, all audiovisual components and all external connections to other devices like the Phantom Premium haptic device and joysticks. Models were modified by adding new building blocks that were made as a part of this project to the model and modifying the existing model to work with these new components and devices. Below is an example screenshot of one of the models opened in virtools (Fig.2.28).

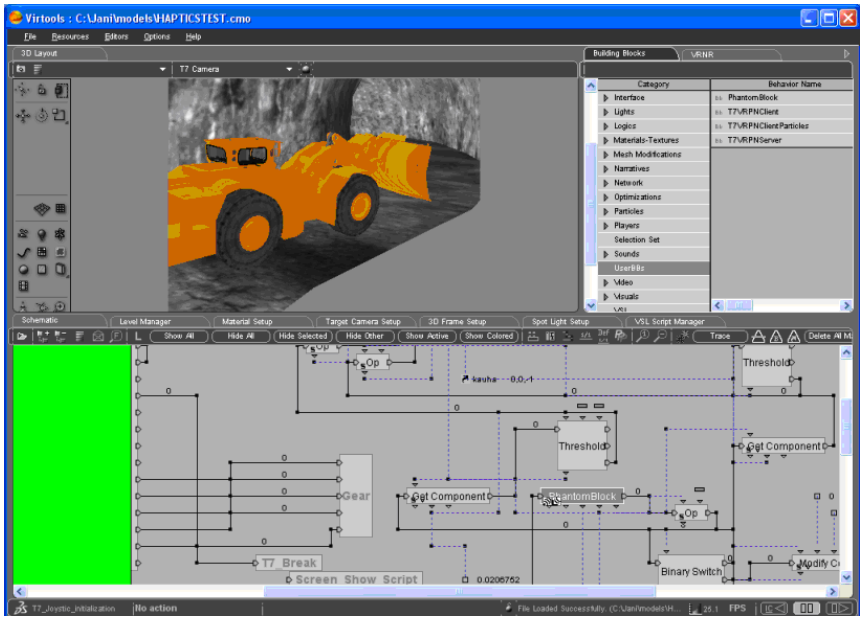

Fig.2.28 Sandvik loader model opened in Virtools

Originally 2 models were obtained from VTT. One of the models was for controlling the loader with a keyboard and the other was for controlling with a gamepad. The gamepad version was modified to the two versions mentioned earlier. First version was modified to use two joysticks and included also few additional features. In this model user could use one joystick button as a kind of a parking brake. Also some other simplifications and modifications like compensation of

joystick's dead zone were made to make the model easier to drive for demonstration purposes for people without experience in driving a machine like the Sandvik loader. The bucket movements of the loader were to be controlled with one joystick and backwards, forwards and steering movements were done with the other joystick. In the following, the modifications to the VTT models and also some basics of the functionality of the behavioural model will be explained. In the Fig.2.29 below the main parts of the model program can be seen.

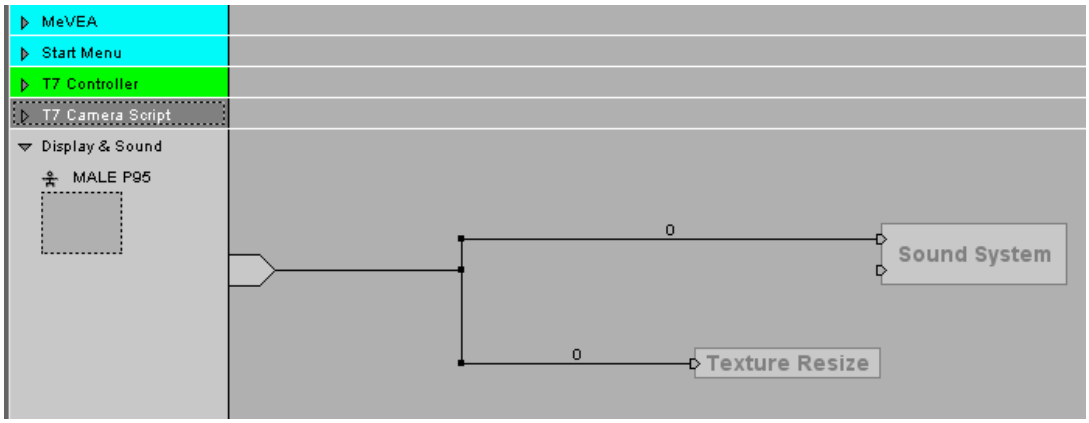

Fig.2.29 Main menu of a Virtools model graphical program scripts

On the left of the picture is a list of the different program scripts, each associated with functionality of their own. The first in the list is MeVEA which contains the functionalities of transferring information from and to the external dynamics solver program provided by Mevea Oy. Next in the list is the Start menu which includes things to do at the start-up of the model when playing mode is started. The T7 controller contains all the functionality related to joysticks and other things that define how user commands affect the model.

T7 camera script includes functionality and definitions for a camera that the user can pan and fly with joystick around the loader to get a birds eye view of the place. Display and sound contains settings and features of the display and sound environment. In this model it contains behaviours of different sound sources and their locations in the 3d model etc.

When double clicking the T7 controller part of the list of program scripts the following view is opened (Fig.2.30).

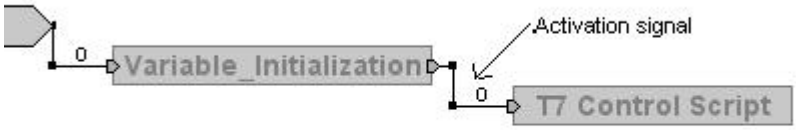

Fig.2.30 Two blocks in the T7 Controller script

These blocks are kind of subprograms into which the user can make a logical group of functions to make reading the program easier. The horizontal connector between the blocks is an activation signal. It starts from the left and by connecting the blocks in the desired order, the user can define in which order the blocks are executed or if they are executed at all. When clicking the T7 Control Script with mouse the following view is opened to the user (Fig.2.31).

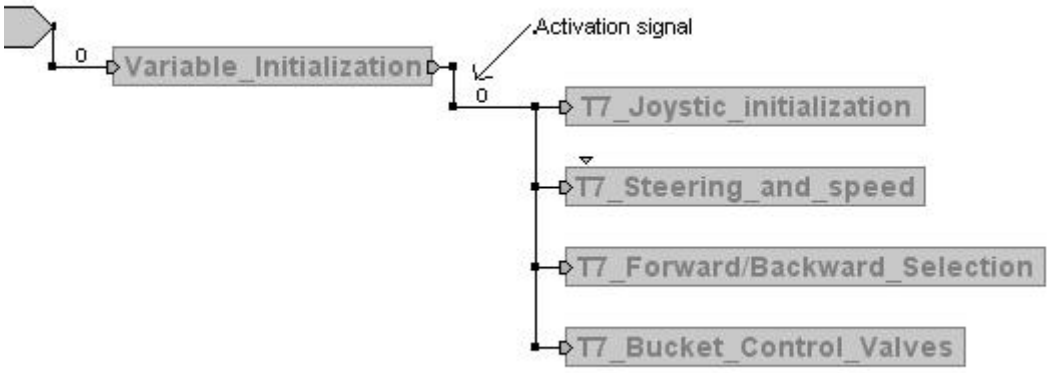

Fig.2.31 Blocks in script

Again more blocks are opened. This part is still the same in both versions of the models. Joystick_initialization contains not only the initialisations, but also most of the functionality of the joysticks. Steering_and_speed block translates and adapts the steering and speed related commands from joysticks to a suitable behaviour of the loader. The same idea is also behind the other two blocks.

In the Fig.2.32 is the view of the insides of the Variable_Initialization block.

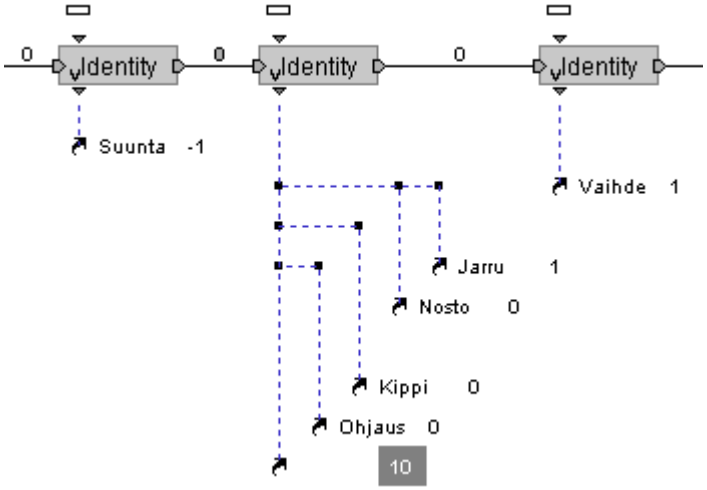

Fig.2.32 Inside the Variable_Initialization block

The identity block assigns a numeric value to a “global” parameter variable. The parameters in this block are all connected to the behavioural model of the loader and operate it’s functions. Vaihde is the number of current gear, Jarru is the state of the loader brakes, Nosto tells the direction and speed of the bucket up/down movement, Kippi represents the speed and direction of the loader’s bucket tilt movement. Ohjaus is the parameter to control the steering movements of the loader and the last parameter is the value controlling the motor of the loader. Parameter Suunta describes the forwards/backwards movement direction of the loader.

When opening the Joystick initialisation block, a view opens with the following picture (Fig.2.33) in the left side of the view. The total view is split into two parts in this document to make it easier to understand the different functionalities as there are several parallel execution threads in each program block.

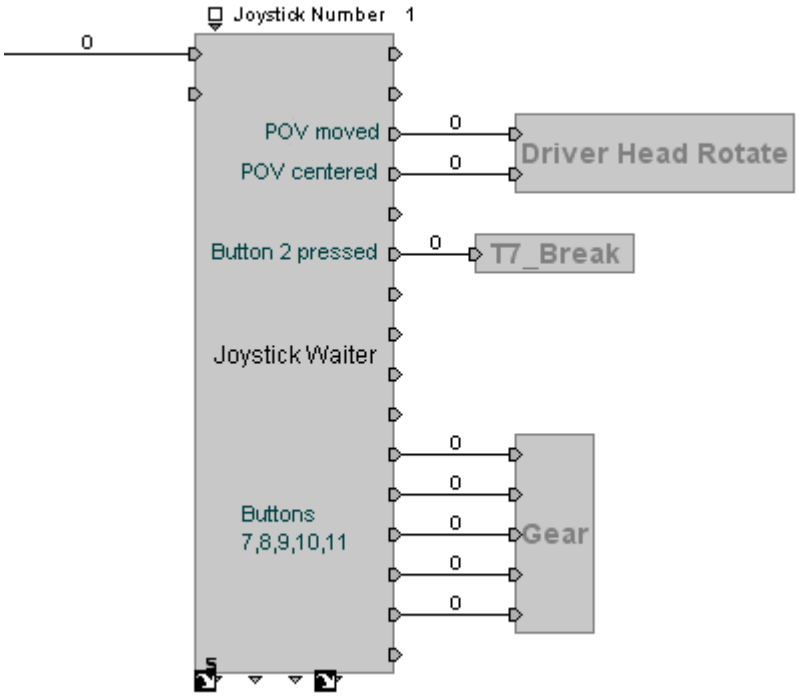

Fig.2.33 Joystick waiter block for joystick 1

The Joystick Waiter block interfaces the model to a joystick attached to the computer. This block is configured for the one of the two joysticks. In this block the joystick number is 1. When someone moves a joystick, from this block the information of the amount of the movement or just the event itself can be obtained or the block can output signals depending on which of the joystick buttons is pressed. The connectors on the right side of the block are activation signals that can be used to activate other blocks. In the block in the picture these activations occur when the small joystick on top of the main joystick is moved or centred, or when any of the buttons 2, 7, 8, 9, 10 or 11 are pressed. From the connectors below, one can get the vector values containing the absolute and relative movement values of the joystick for different axes. These values are used for the steering and movement control of the loader. The other blocks which are activated by this Joystick waiter block are the Driver Head rotate block which takes care of the panning and zooming of the a camera around the loader, button 2 puts on the “parking brake” block of the loader and buttons 7 to 11 change the gears.

In the next Fig.2.34 is the T7-brake block that takes care of the parking brake feature of the loader simulation.

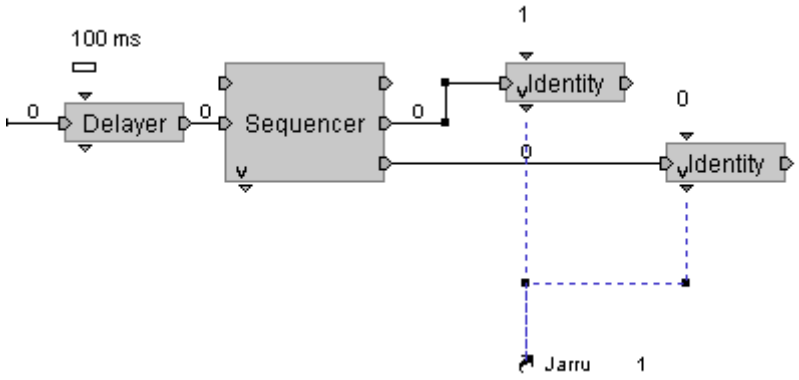

Fig.2.34 T7-brake block

The idea behind this block was to provide a brake that is on when pushing a button and when the same button is pushed again, it is off and so on. As the process of going thru these blocks, when running the simulator, occurs several times a second, these blocks would go thru the on/off sequence several times in the time that it takes for human to press a button. This would lead to the situation that the state of the button and the brake would be almost random. For correcting this problem, there is the Delayer block that makes sure that only one push of the button is recorded for 100 ms time, guaranteeing the button state is read only one time per push of the button. The value of 100 ms was found by experimentation. The Sequencer block activates one of its outputs at a time for every push of the button in a sequence starting from the upmost connection and going downwards until starting again from the beginning.

As the button is pushed, the sequencer activates only one of its' two outputs at a time activating the connected Identity block that sets a numeric value to the Jarru (brake) variable that is connected to the dynamic model of the loader. Jarru variable represents the state of the brakes of the loader, 1 is brakes are on and 0 means that brakes are off.

Fig.2.35 is the second half of the insides of the Joystick initialisation block.

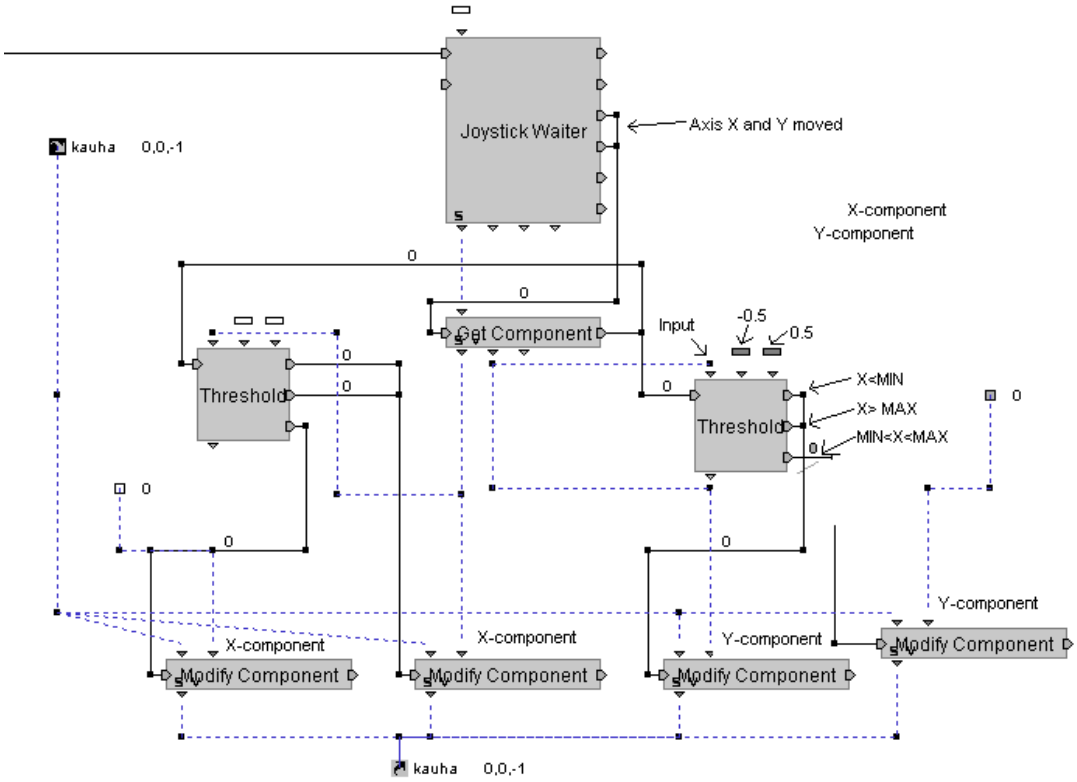

Fig.2.35 The loaders bucket functionality with the second joystick

In this part there is the same Joystick Waiter block, but it is configured for the second joystick with number 0. This block is for the up/down and tilt movements of the loaders bucket (kauha). The output values of the Joystick Waiter block used are the relative movements of the joystick and all the blocks are activated by the x and y axis movements of the joystick. Threshold blocks check if an input value is between two user defined values Min and Max. Depending on the result, different outputs are activated. If the values are outside the defined range they are output to the Modify component blocks where they are placed into the correct components of the parameter vector kauha that defines the bucket movements. If the values are between the defined min and max values, zeros are put into the values of the vector kauha stopping the movement of the bucket. This functionality was made to create a big enough dead zones as the centring of the joysticks was not very good and a zero could not always be guaranteed to stop the movements of the bucket.

The second modified model was to be used with one joystick and the Phantom haptic device. Joystick controlled the backwards and the forwards movements and braking of the loader. Haptic device was used for the loading movements with the bucket so that user takes a hold of the stylus like a small shovel and operates the bucket like shovelling with it. In Fig.2.36 is the Phantom stylus where the movements for the bucket can be seen.

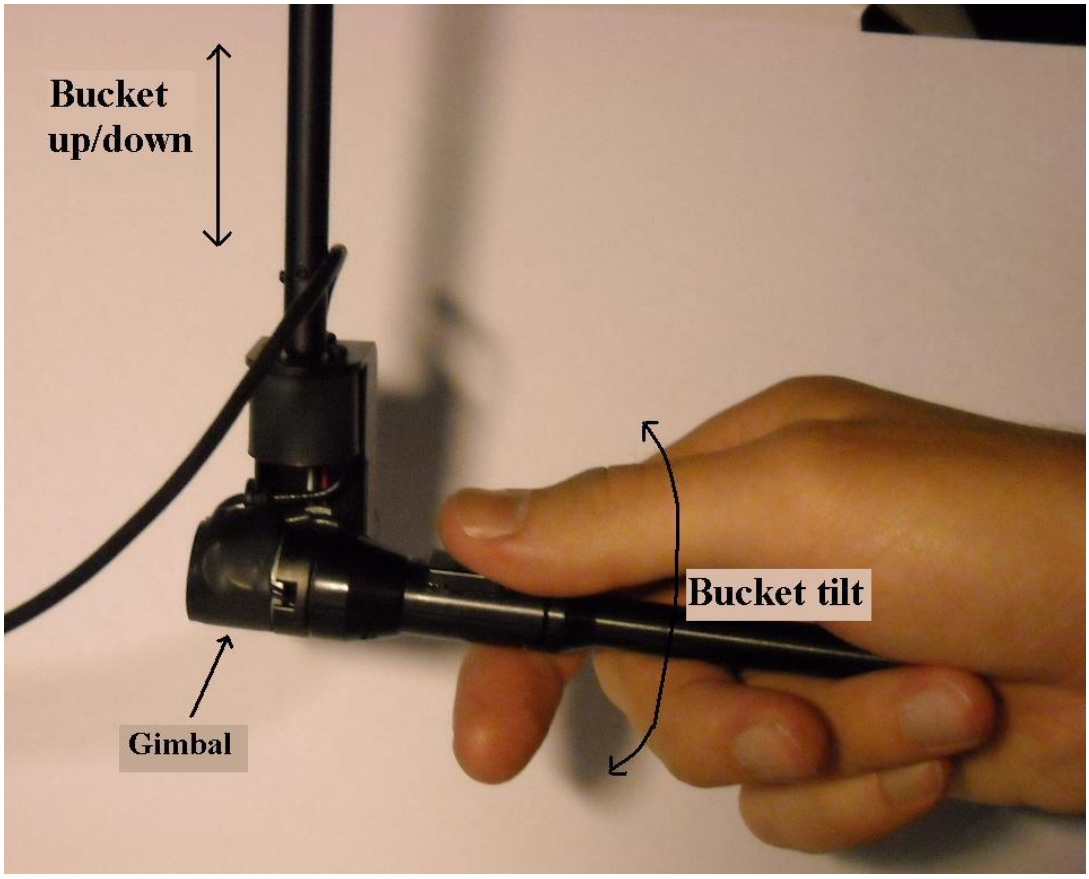

Fig.2.36 Bucket movements with the haptic device

When keeping the button on the stylus pushed down the movements of the haptic device were transferred to the virtools model. Sideways movement of the stylus and the gimbal operated the steering of the loader letting the user to control the direction of forwards and backwards movement. Up and down movements of the stylus moved the loader bucket up and down. Tilting of the bucket was done with the tilting of the stylus.

In Fig. 2.37 is shown the steering movement of the haptic device.

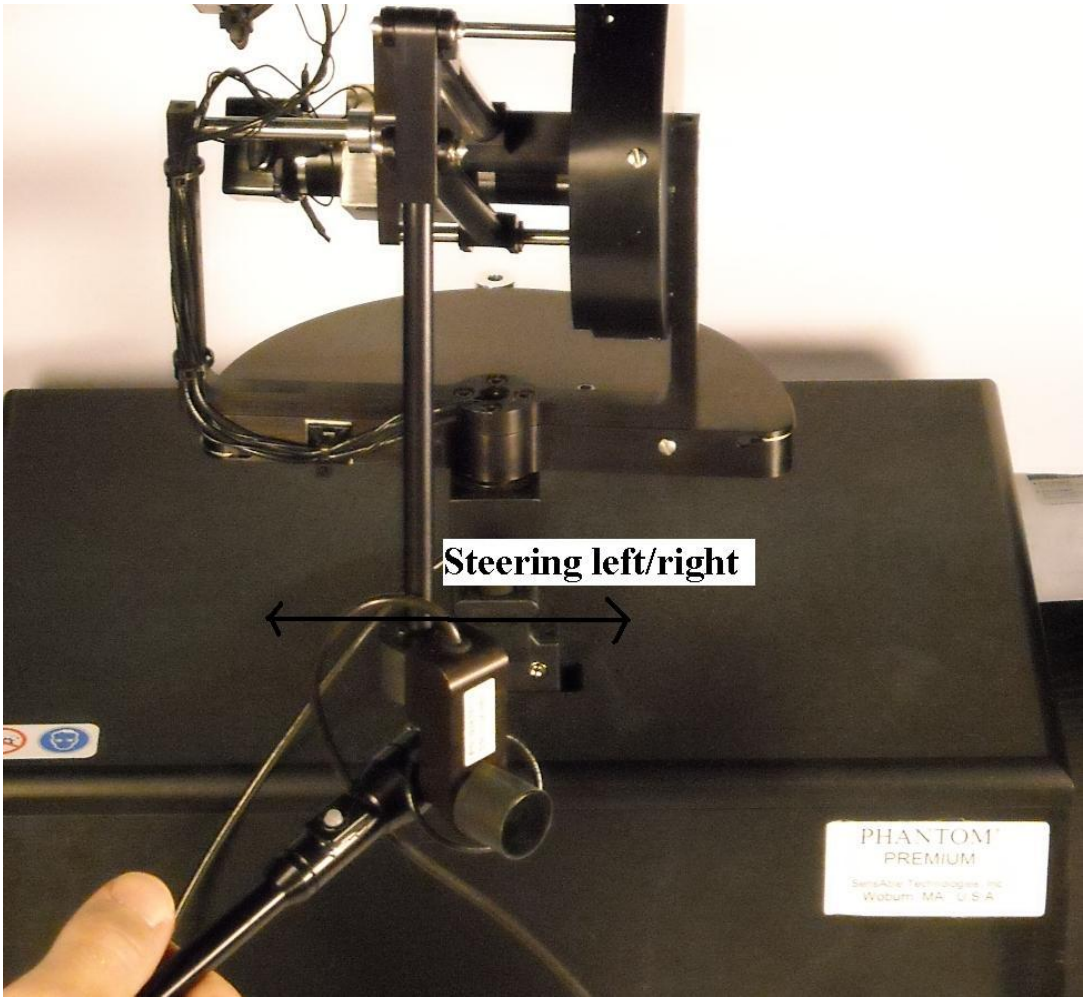

Fig.2.37 Steering movement with the haptic device

Originally the force feedback features of the Phantom were supposed to be used with this model, but due to time limitations this was postponed to be done in future research projects. As the movements of the bucket and steering meet their endpoints or hit some obstacle like a stone, there would have been the equivalent stop of motion and force felt on the haptic device and on the user's hand.

In Fig.2.38 are described the modifications made to the model of the loader to implement the Phantom haptic device for the control of steering and the bucket.

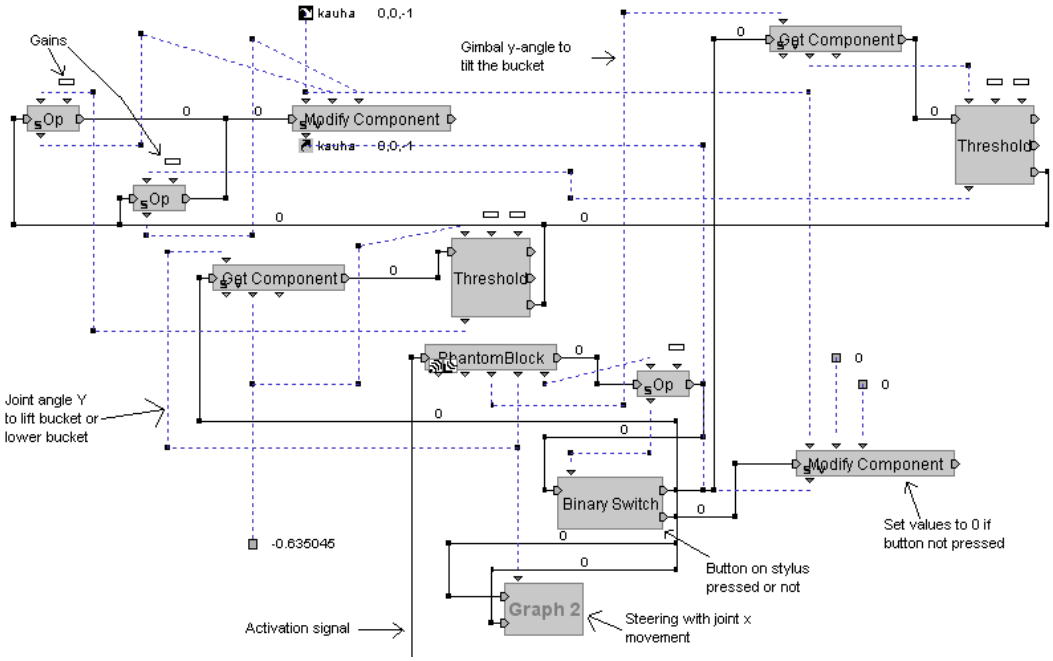

Fig.2.38 Functionality of the haptic devices control of the loaders bucket

In the middle of the Fig.2.38 is the PhantomBlock, which connects to the PhantomVRPNServer program using a VRPN connection. The Binary Switch block activates one of its outputs depending on if the condition set to the input is fulfilled or not. In this model this block activates one input if the PhantomBlock outputs 0 and the other if it PhantomBlock outputs 1 from the output representing the state of the button on the stylus. The Op block does to one or two input values what ever operation defined inside the settings of the block. In this model the input is just one numeric variable on which the block executes abs function to take the absolute value and then outputs this value.

In the next Fig.2.39 is a flowchart describing the part of the model in previous picture with more details.

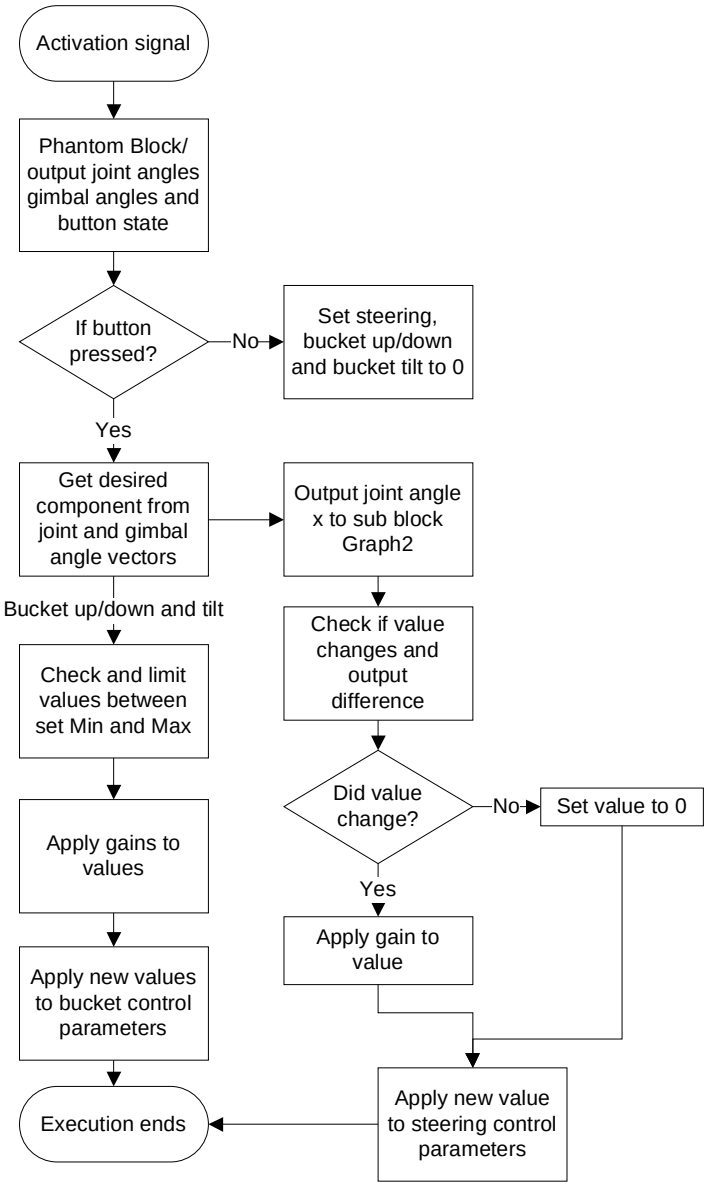

Fig.2.39 Flowchart of haptic device control of the bucket and steering

When receiving the activation signal, the Phantom block outputs the joint and the gimbal angles and the stylus button state of the haptic device. If the button is not pressed, all the values related to the bucket control and steering are set to 0. If the button is pressed, the desired components of the output vectors from the Phantom block are extracted. The bucket related values are then checked against maximum and minimum values and gains are also applied to the values. Then the bucket

control values are output to the actual “global” control parameters. If the movement of the haptic device is a steering movement, the vectors from the PhantomBlock are directed to another block (Fig.2.40). Inside this block it is observed if the values have changed from previous values. If the values have not changed, the output values for steering are set to 0. If the values have changed, the user has made a steering movement with the haptic device and a gain is applied to the values after which the values are output to the control parameters of the loaders steering system.

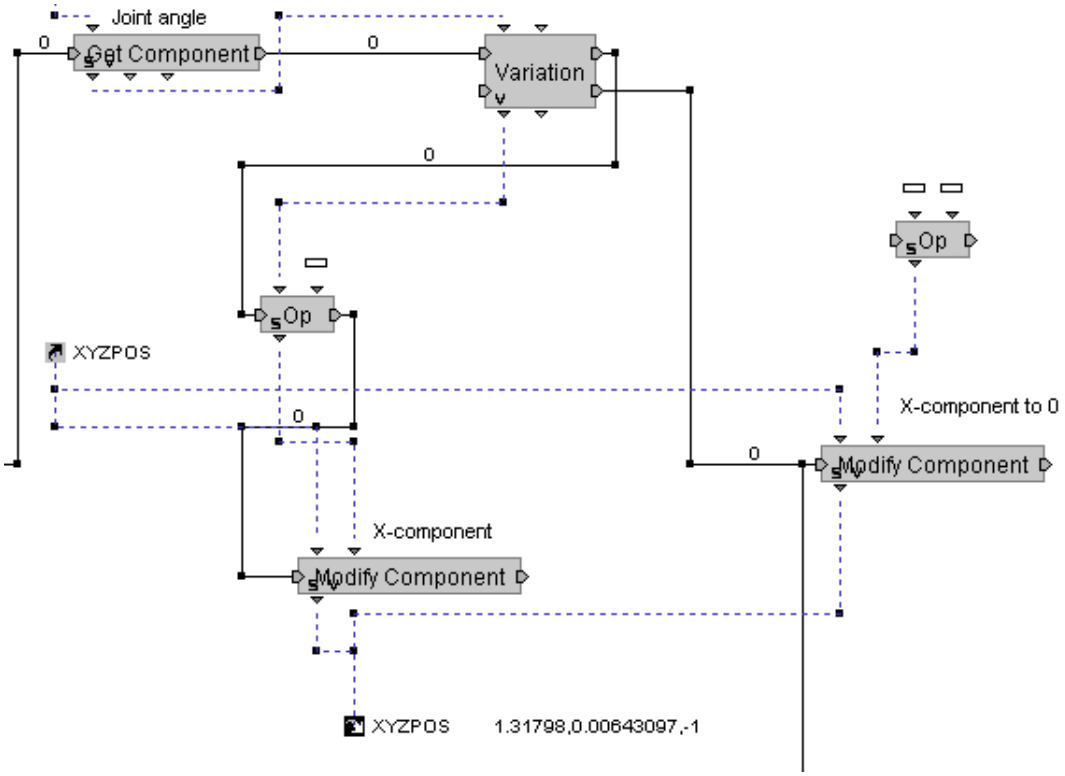

Fig.2.40 Steering of the loader with the haptic device

The Variation block in the above picture activates its outputs depending if the input value has changed or not. If value has changed, it also outputs the difference between the previous and the current values. With this block, the absolute movement of an object can be converted to relative movement.

3 THE CONSTRUCTION PROCESS OF THE SIMULATOR ENVIRONMENT

Construction of the simulator environment was done in several parts, testing each part individually and also when attached to other components of the system. Main concern with the motion platform and its controller was the safety of both the users and the device, as the platform had broken down several times before due to various reasons. Software components were mainly made using Microsoft Visual Studio 2005 programming environment, Matlab/Simulink and dSpace Control desk graphical user interface building program.

3.1 Software

None of the software made for this project had the need to be hard real-time software, as itself using Windows as the platform for the programs made it impossible to have a completely deterministic system. Also as Virtools and Mevea's solver were running on Windows and the Phantom haptic device was connected thru PC's parallel port, there was not much to be done with some timing problems and delays (Zhang 2005). Main concentration was thus on the security of the software components, that they could handle unexpected situations and events and also be usable despite of inaccurate timing and unpredictable delays inherent to Windows. Also in the beginning of the process it was unclear whether all the software components of the system could be run on a single computer and still achieve reasonable level of reality in the simulation.

All the separately run programs in Windows also had to be made so that when other programs crashed, the simulators programs would not be affected so that a controlled shutdown of the system was still possible. Any uncontrolled movements of the Stewart platform could potentially be dangerous to anyone near the device or sitting on top of it. Both Solver2RS232 and PhantomVRPNServer were realised with separate execution threads, one thread for each the window, the user interface and for the actual part of the program that did the communications and data encoding functions.

In the beginning, several different versions were made before the latest version of the Solver2RS232 program. These versions were done to test different ways to control the Stewart platform. First the platform was just tested with a simple Matlab/Simulink model running inside the dSpace computer.

After that an attempt was made to run the controller for the motion platform inside a program running on the main computer with Windows XP as operating system. This version would have used an analogue/digital I/O-card and a pulse counter card for interfacing to the servo modules. After few attempts, this version was given up, as the safe use of the Stewart platform could not be guaranteed with this method. Next version was to use the main computer with the dSpace computer to control the motion platform. The only difference to the final version was that instead of RS232, 6 analogue signals were used in transferring the position and the orientation reference information to the dSpace computer. However due to calibration problems and because of too big a risk with induced errors into the signal pathway, this version was changed to use RS232 serial connection instead. Initially in RS232 transfer, a different method of encoding the data was used. It included transferring the data as ASCII characters to the embedded computer, but because of the memory buffer size limitations of the dSpace computer, the encoding method explained earlier in previous chapter was chosen.

Testing of the Solver2RS232 software was first done with a normal computer replacing the dSpace computer to check the integrity of the data in as many situations as possible. Testing was done with Windows own terminal program HyperTerminal. After this the program was tested with all the parts of the environment running, but without anyone sitting on the motion platform.

After driving around the modelled environment of a mine in the demonstration simulation of the Sandvik loader and after also trying to cause different extreme movements with the loader, the gain values were adjusted by estimation on how rough the movements were. Only when all people working on the platform agreed that it was safe, tests with a human sitting on top of the platform were started. During these tests, only the accelerations, maximum lengths of some movements

and coordinate system positioning was adjusted and changed to be more suitable for demonstration purposes for the inexperienced users. In more realistic levels the movements could sometimes be too rough for some users.

With the PhantomVRPNServer program, the main target was to realise the program so that the user could get all possible information out of the haptic device in Virtools as fast as possible, without interfering the other programs in the main computer. Verifying of this feature was done by testing of all the software components running at the same time and it seemed that the structure of using parallel executing threads was enough to reach this target. Reason for making this program was that the most common use of haptics in VR environments is to use it as a manipulator or as a sort of a pointer device like a mouse in the virtual 3D space, but in this project it was to be used like a conventional joystick with more then 2 degrees of freedom.

PhantomVRPNClient building block had initial requirements of being usable like the joystick building blocks in Virtools, to provide a continuous flow of data and to provide also information of the status of the button located on the haptic device's stylus.
Main frame of the program was created with the help of a wizard program provided with Virtools. According to the manual of the Virtools software development kit, this wizard required the use of Microsoft Visual Studio 2003 programming environment. As the other programs were made in Visual Studio version 2005, this building block software project was merely created in the earlier version of Visual Studio using the provided wizards and then converted to be programmed in Visual Studio 2005. Testing of this building block was carried out first in a separate model without any other features and next it was tested in the full model of the loader.

Program for the dSpace computer was done in Matlab/Simulink program using the ready made blocks provided by dSpace. In the beginning of testing the Stewart platform a simple P-controller made by Professor Huapeng Wu was used with inverse kinematics to control the motion platform in both open loop and closed loop modes. With this program the usable movement limits for the platform were

estimated. After the decision to use the dSpace computer for the final version of the motion platform controller, the new version of the Simulink model was built on top of this early version. Components for receiving the analogue signals, handling RS232 serial communication and decoding of the serial data were added to the model and tested separately without running the platform. Also the digital output for enabling and disabling the motors was added at this stage for safety reasons as initial conditions of servo modules could not always be guaranteed without this feature. Several security limits and gains were added to the simulink model and the model was tested in dSpace computer with the full system, including the motion platform.

3.2 Hardware

Hardware requirements in this real-time simulator environment were mainly related to the controller used with the motion platform and the main computer running the Windows based programs and Virtools. For the main computer the requirements were to have a minimum of CPU with 2 cores to enable true parallel processing, 3D-surround soundcard to make future use of 3D spatial sound environment possible, serial, parallel and ethernet connections, analogue/digital I/O card, pulse counter card, video card capable of running Virtools 3D environment and minimum 3 to 4 USB connections. Other requirements were based on the hardware requirements of Microsoft Visual Studio and the Virtools program.

For the main controller of the motion platform, the most important requirement was safety and reliable real-time operation. Also there was a need for analogue and digital I/Os, pulse counter functionality, RS232 serial port and ethernet.

3.2.1 Meeting the requirements with available hardware

A PC which met or exceeded the requirements was already available in the laboratory where the environment was to be built; only installation of the required software was needed. Also an embedded dSpace computer that met most of the requirements for the motion platform controller was available for use in this project.

3.2.2 Problems

With the dSpace computer there was a problem that the ethernet connection included on the main processor card could not be used for users own programs and therefore the possibility of testing the connection between solver and the motion platform via ethernet was excluded from the project.

In the main computer the sound system did not always function properly with the available Virtools model for the Sandvik loader, but this could also be due to configuration problems with the sound card.

When having to update more then 2 camera views on the screen at the same time the update rate of the graphics became disturbingly slow. However this did not seem to slow down the operation of the other programs and devices in the computer.

4. USER TESTING OF THE SYSTEM

4.1 Test plan

The simulator environment was to be tested with drivers with and without previous experience in driving the Sandvik loader. During the testing, the users were allowed to comment freely as the tester observed their physical response to the simulator environment and asked questions related how realistic they thought the systems was and what could be improved.

The level of immersion can be estimated with grading the used methods and devices in the simulator environment, but presence estimation is still based on user feedback as to being very subjective in nature (Bowman 2007). Therefore a more objective method of measuring the realism should be used in the future, but for this limited first test, there was no need for other then asking simple questions from the user.

4.2 Results

According to the testers who had not used a loader before, the experience was very realistic in the sense that the platform movements occurred almost synchronously with the movements in the image of the model projected to the screen. Some movements were not considered rough enough and could be with higher acceleration levels. As the users practised driving the loader for the first time with the simulator, it was noted that they were physically tired and that driving was physically demanding. This can also occur with a real machine when extreme concentration is needed for learning something new. Also the effects on body in the form of the movements and accelerations produced realistic physical strains and muscle pains when the user tried for long time to keep his body in control in the very rough ride of the loader inside the mine.

The people working on the project also tested the environment with higher accelerations and higher realism levels. In a few accidental crashes into a wall of the mine, the simulator gave an extremely realistic and painful feeling of collision as the drivers flew against the four point seatbelts. Often when starting the simulation, the user expected the simulator to behave more like a game where

nothing bad can happen and no pain, tiredness or strain can be felt. But when for the first time that some part of the loader hit some object or wall or when the loader made some other harsh movements, the user woke up in realisation that it is no longer a game, but extremely real. After this realisation the users started to be much more accurate in driving and concentrate much more as if driving a real device. If setting the realism level to one the higher levels possible for the motion platform, such high accelerations can be achieved that they can already produce very serious physical damage to the user if in collision, just like in a real mobile machine.

User with previous experience in driving the loader on normal smooth surface commented mainly on the aspects of how to make the controls of the loader simulator i.e. the joysticks, sound environment, displays around the motion platform and the seat itself more realistic if this environment were to be used as a training device for the loader.

Testing of the haptic device was done only with a very small amount of testers and only on screen without anyone sitting in the chair on the motion platform. This was because of time limitations not allowing installing the haptic device to the motion platform safely.

In the tests, the following things were observed from using the haptic device in controlling the bucket like a shovel. Using all 6 DOF for movements of the machine seems to be too much for a human to control. When reducing the used movements to only 3 DOF and one button, it still is difficult but can be learned. Main problem of the difficulties with 3 DOF was that the movements of the loader were not fast enough to respond the way human expects when using a shovel thus causing a need for the human to slow down responses to the level of the machine.

Also during the tests it became apparent that for more frequent use of the simulator environment, improvements need to be made to the operator user interface of the motion platform. For motion platform there should also be an automatic initialisation procedure that drives the platform into 0 position with just one push of

a button. Currently the amount of functions and complexity of the platform controller is good for testing purposes, but too much for more frequent use.

4.3 Conclusions of simulators applicability

From the users feedback it can be noticed that the motion platform gave a very good feeling of driving a real device, although a big part of this is also because of the dynamics solver by Mevea as the platform just does movements according to the instructions from the solver. It can be said that this project succeeded in synchronising the components together. Also the recommendations on how to modify the system to make it more realistic for training purposes, can be easily realised to the current system as well as making the motion platform user interface more easier to operate. This shows that the system also achieved the purpose of being easily adaptable for different simulation purposes thanks to its modularity and the 6 DOF motion platform.

The simulator system tests gave also a warning that the level of realism now reachable can also be dangerous if not controlled properly. There needs to be serious discussion and research on what level of realism is actually needed and how to prevent human injuries with the most realistic levels of simulation and still maintain the level of reality for things that are considered safe.

The use of haptics in the simulator environment can also be adapted to many needs and the results also show that the intended control method for the bucket of the loader with haptics could also be usable with some modifications to the models and the haptic device.

5. FURTHER RESEARCH AND DEVELOPMENT

5.1 Improvements for the simulator environment

Below are suggestions on how to make the current version of the simulator environment more user-friendly, safer and add the level of realism for other parts then the motion platform.

1) The dSpace controller could either be updated or replaced with a smaller sized device with an ethernet connection, more security features and with a possibility to program it with a sequential programming language like c++. Also a requirement for the controller is the possibility to be able to program the device using the freely available VRPN libraries so it could be connected to the dynamics solver without any conversion programs in the middle.
2) Adding sensors to the motion platform to make possible completely automatic initialisation procedure and reference point driving features.
3) Updating of the main computers display cards and obtaining a license from Virtools to be able to use multiple camera views at the same time on different screens.
4) Higher quality wireless joysticks with full force feedback for all axes.
5) Attaching haptics to a movable platform with wheels, computer and wireless network to enable easier use in various locations in laboratory (Fraser 2008)
6) Addition of a VR glove, possibly with haptic feedback and a head mounted display to the laboratory equipment as these would give better immersion level if simulator would be used for prototype testing of user interfaces.
7) Quick attachment system on top of the motion platform to allow easier and faster changes and modifications to mock-up on top of the platform. Also stairs should be built to allow easier access to the platform.
8) Adding mechanical brakes to the motion platforms linear actuators to keep the platform still when a user is climbing onto it.

5.2 Possible uses of the simulator environment

Below are some suggestions on the numerous possible uses of the real-time simulator environment.

- Development of training systems
- Testing of 3D models and dynamic solvers.
- Design prototyping
- User interface testing and development
- Recording and research of user habits and actions when driving a vehicle or using a device/machine.

5.3 Future research possibilities

Research based on psychomechatronics (Yousefi 2009) applying cognitive methods on the whole machine design process and also including the use of autonomous operation, teleoperation and telepresence with virtual and augmented reality in the man machine interface.

Often new mobile machines are just updates of older versions and they are results of optimising what already exists thus resulting in compromises when it comes to the machine performance. Also very often compromises are due to the fact that the human and machine are physically in contact with each other and the limits of human body endurance and capabilities are limiting the performance of the machine as a whole. As teleoperation, virtual and augmented reality and autonomous systems have made significant advances lately; this limiting factor of physical connection between human and machine can be removed, modified and optimised to take the full advantage of both human and the machine (Stemmer 2004). Design targets and design itself can be optimised separately for both machine and human and also at the same time for the combination of both of them. In Freund 2001, is described the possibility of using a method called task deduction for removing the human operator from the real-time loop of the machine. While the user is working in a virtual reality environment, a planning component is used to split the work tasks into smaller basic tasks that are then carried out by the machine. This way the machine can operate in the most optimal way for the machine without the human limitations and also without the delays and the need for high accuracy in the virtual

environment can be removed. If the mobile machine and its control system are versatile enough, there is no need for the user to even see the machine; the user gets the impression of performing the task by himself which is the highest level of immersion and naturality of control that can be achieved. This method of control using a robot in space in Freund 2001 was tested very successfully on a real space mission by German Space Agency, Japanese Space Agency and Institute of Robotic Research in Germany.

6. CONCLUSION AND RECOMMENDATIONS

Target of the project was the integration of a real-time simulator, a motion platform and a haptic device. Project was a success as the real-time simulator environment was built successfully and featuring modularity that enables the environment to be modified to whatever need desired.

All the components of the system worked in good synchronisation so that the users got a “realistic” enough experience in driving the Sandvik loader despite using Windows as platform for most of the program components. Security of the motion platform was also noted as a very important topic that still needs improvement. As the level of reality especially in the motion platform has reached levels that could be dangerous to the user, there needs to be serious consideration on how real things should actually be and how to make sure no accidents happen.

Recommendations were also given, on the possibilities of use for the environment and on what modifications need to be made to make the environment safer, more realistic and user friendly.

Integration and testing of the Phantom haptic device in the simulator environment was done successfully and it was noted that the use of a haptic device in controlling a machine is possible, but more research is needed in this subject.

With this simulator environment, research can also be made to change the way the initial targets of mobile machine design are set. The design can be changed from a compromise between human and machine to a fully optimised design for both separately and together by removing the human operator from the real-time loop.

REFERENCES:

Barca Jan Carlo, Li Raymond Koon. Augmenting the Human Entity Through Man/Machine Collaboration, Berwick School of Information Technology, Faculty of Information Technology, Monash University, Melbourne, Australia, [e-document] 2006, [retrieved May 29, 2009]. From: http://ieeexplore.ieee.org

Blake Jonathan, Gurocak Hakan B, IEEE/ASME TRANSACTIONS ON MECHATRONICS: Haptic Glove With MR Brakes for Virtual Reality Member, [e-document] 2009, [retrieved May 29, 2009]. From: http://ieeexplore.ieee.org

Bowman Doug A, McMahan Ryan P, Virtual Reality: How Much Immersion Is Enough? Virginia Tech, [e-document] 2007, [retrieved May 29, 2009]. From: http://ieeexplore.ieee.org

Bracewell R. Martyn, Wimperis Andrew S, Wing Alan M. Brain Mechanisms of Haptic Perception, University of Wales Wolfson Centre for Clinical and Cognitive Neuroscience Bangor, University of Wales, UK, Behavioural and Brain Sciences Centre, University of Birmingham, [e-document] 2008, [retrieved May 29, 2009] From:http://www.springerlink.com

Carlton University, Human Oriented Technology Lab, [retrieved June 5, 2009] From: http://www.carleton.ca

Cole Johathan, Dinse Hubert R, Andersen Peter A, Ballesteros Soledad, Hatwell Yvette, Gentaz Edouard, Grunwald Martin. Human Haptic Perception: Basics and Applications. Birkhäuser Basel, [e-document] 2008. ISBN 978-3-7643-7611-6 (Print) 978-3-7643-7612-3 (Online), [retrieved May 29, 2009] From: http://www.springerlink.com

Connell Andrew. Reality Finally Bites, Virtalis Group, [e-document] 2005, [retrieved May 28, 2009] From: http://ieeexplore.ieee.org

Craig, John J. Introduction to robotics : mechanics and control, Upper Saddle River (NJ). Prentice Hall. 2005. ISBN

Dipietro Laura, Sabatini Angelo M, Dario Paolo. A Survey of Glove-Based Systems [e-document] 2008, [retrieved May 29, 2009]. From: http://ieeexplore.ieee.org

Fraser Mike, Duff Paul, Pearson Will, Chapter: Grounding Mobile Force Feedback in the Real World, Department of Computer Science, University of Bristol, Book: Haptics: Perception, Devices and Scenarios, [e-book] 2008, [retrieved May 29, 2009]. From: http://www.springerlink.com/

Freund Eckhard, RoBmann Jurgen. Multimedia and Virtual Reality Techniques for the Control of ERA, the First Free Flying Robot in Space, Institute of Robotics Research (IRF), Germany, [e-document] 2001, [retrieved May 28, 2009] From: http://ieeexplore.ieee.org

Gunzelmann Glenn, Lyon Don R, Mechanisms for Human Spatial Competence Air Force Research Laboratory, L3 Communications at Air Force Research Laboratory ,Mesa, AZ, United States, T. Barkowsky et al. (Eds.): Spatial Cognition V [e-document] 2007, [retrieved May 29, 2009] From:http://www.springerlink.com

Handroos, H. Mekatroniikka. Lecture notes. Lappeenranta University of Technology

Hayes James, virtual reality enters the real world, [e-document] 2008, [retrieved May 29, 2009]. From: http://ieeexplore.ieee.org

i-O Display Systems, [In company products] 2009, [retrieved June 5, 2009]. From: http://www.i-glassesstore.com/

Katsis Christos D, Katertsidis Nikolaos, Ganiatsas George, Fotiadis Dimitrios I. Toward Emotion Recognition in Car-Racing Drivers:A Biosignal Processing Approach, [e-document] 2008, [retrieved May 29, 2009]. From: http://ieeexplore.ieee.org

Laplante Philip A. Real-Time Systems Design and Analysis, Institute of Electrical and Electronics Engineers [e-document] 2004, [retrieved May 29, 2009]. From: http://ieeexplore.ieee.org ISBN 0-471-22855-9

Mathworks Inc. [In Mathworks examples and tutorials]. 2008. [retrieved June 5, 2009] From: http://www.mathworks.com/

Noreils Fabrice R, Adding a Man/Machine Interface to an Architecture for Mobile Robots, Alcatel Alsthom Recherche, France [e-document] 1991, [retrieved May 29, 2009]. From: http://ieeexplore.ieee.org

Novint Technologies Inc. [www-document] 2009. [retrieved June 5, 2009] From: www.novint.com

Pasquini A, Pistolesi G, Risuleo S, Rizzo A, Veneziano V. Reliability analysis of systems based on software and human resources, University "La Sapienza", Rome, Italy, Dept. of Communication Science, Univ. of Sienna, Sienna, Italy, Centre for Software Reliability, London, United Kingdom, [e-document] 2001, [retrieved May 29, 2009]. From: http://ieeexplore.ieee.org

Pearson Ian, our virtual future, [e-document] 2008, [retrieved May 29, 2009]. From: http://ieeexplore.ieee.org

Pellinen Pekka, Haptiikkalaitteen hankinta ja käyttöönotto, Department of Mechanical engineering, Lappeenranta University of Technology, [e-document] 2008, [retrieved May 28, 2009] From:www.lut.fi

Sallinen Joni, Liikealustan suunnittelu liikkuvan työkoneen simulaattoriin, Lappeenranta Technical University, Department of Mechanical engineering, [e-document]. 2008, [retrieved May 29, 2009]. From http:www.lut.fi

Sharit Joseph, A Methodology for Promoting Reliable Human–System Interaction, Department of Industrial Engineering, University of Miami, Florida, USA, [e-document] 2008, [retrieved May 29, 2009] From:http://www.springerlink.com

Stemmer Ralf, Brockers Roland, Siegbert Drue, GET Lab, University of Paderborn, Germany, Thiem Jorg, Hella Hueck KG, Lippstadt, Germany, 2004 IEEE International Conference on Systems, Man and Cybernetics: Comprehensive Data Acquisition for a Telepresence, [e-document] 2004, [retrieved May 28, 2009]. From: http://ieeexplore.ieee.org

Tsai Lung-Wen. Robot Analysis: The Mechanics of Serial and Parallel Manipulators. Wiley-Interscience. 1999. ISBN-13: 978-0471325932

Wege Andreas, Technische Universität Berlin, Department of Computer Science, Institute for Technical Computer Science Real-time Systems & Robotics. [www-document] 2009. [retrieved June 5, 2009] From: http://pdv.cs.tu-berlin.de/HandExoskeleton/HandExoIntro.html

Wu Huapeng, Introduction of modelling, design and control of parallel robot, Lecture material, Lappeenranta University of Technology [e-document]. 2008

Yousefi Hassan, Soleimani Amir Mohssen, Handroos Heikki. Human Centered Design of Mobile Machines by a Virtual Environment, Institute of Mechatronics and Virtual Engineering, Faculty of Tech., Lappeenranta University of Tech, Finland, [e-document] 2009

Zhang J, Lumia R, Wood J, Starr G. Achieving Deterministic, Hard Real-time Control On An IBM – Compatible PC: A General Configuration Guideline,

Department of Mechanical Engineering, University of New Mexico, [e-document] 2005, [retrieved May 29, 2009]. From: http://ieeexplore.ieee.org

Zhou ZhiYing, Cheok Adrian David, Qiu Yan, Yang Xubo, The Role of 3-D Sound in Human Reaction and Performance in Augmented Reality Environments, [e-document] 2007, [retrieved May 29, 2009]. From: http://ieeexplore.ieee.org

Zyda Michael. From Visual Simulation to Virtual Reality to Games, USC Information Sciences Institute, [e-document] 2005, [retrieved May 29, 2009]. From: http://ieeexplore.ieee.org