VTT PUBLICATIONS 347

A haptic rendering system for virtual handheld electronic products

Tommi Anttila
VTT Electronics

TECHNICAL RESEARCH CENTRE OF FINLAND
ESPOO 1998

ISBN 951–38–5232–6 (soft back ed.)
ISSN 1235–0621 (soft back ed.)

ISBN 951–38–5233–4 (URL: http://www.inf.vtt.fi/pdf/)
ISSN 1455–0849 (URL: http://www.inf.vtt.fi/pdf/)

JULKAISIJA – UTGIVARE – PUBLISHER

Valtion teknillinen tutkimuskeskus (VTT), Vuorimiehentie 5, PL 2000, 02044 VTT
puh. vaihde (09) 4561, faksi (09) 456 4374

Statens tekniska forskningscentral (VTT), Bergsmansvägen 5, PB 2000, 02044 VTT
tel. växel (09) 4561, fax (09) 456 4374

Technical Research Centre of Finland (VTT),
Vuorimiehentie 5, P.O.Box 2000, FIN–02044 VTT, Finland
phone internat. + 358 9 4561, fax + 358 9 456 4374

VTT Elektroniikka, Elektroniikan piirit ja järjestelmät, Kaitoväylä 1, PL 1100, 90571 OULU
puh. vaihde (08) 509 111, faksi (08) 551 2320

VTT Elektronik, Elektroniska kretsar och system, Kaitoväylä 1, PB 1100, 90571 ULEÅBORG
tel. växel (08) 509 111, fax (08) 551 2320

VTT Electronics, Electronic Circuits and Systems,
Kaitoväylä 1, P.O.Box 1100, FIN–90571 OULU, Finland
phone internat. + 358 8 509 111, fax + 358 8 551 2320

Technical editing Kerttu Tirronen

LIBELLA PAINOPALVELU OY, ESPOO 1998

Anttila, Tommi. A haptic rendering system for virtual handheld electronic products. Espoo 1998, Technical Research Center of Finland, VTT Publications 347. 69 p.

UDC 681.3.07:681.3.06:519.68
Keywords virtual prototypes, virtual reality, 3D modelling, haptic rendering, force feedback

ABSTRACT

A virtual prototype is a fully digital computer model that simulates a product's visual appearance, sound properties, the functionality of the user interface and its behaviour as closely as possible. Touch interaction, however, is one of the main information sources when we explore objects and interact with our environment. For this reason a haptic rendering system that enables the user to receive force and tactile feedback from 3D graphical models is studied. Haptic rendering gives the user the ability to sense the virtual prototype's physical properties such as shapes, materials and button properties. Haptic rendering with a stereographic view provides a realistic and comprehensive simulation of the physical user interfaces of products. An experimental haptic rendering system is developed for virtual prototypes of small-sized handheld electronic and telecommunication devices. The system provides tactile and force feedback from user-to-product interaction through a simulated user interface of the target product. The developed haptic rendering system is based on distributed architecture. The visual rendering process is executed on a separate computer from the haptic rendering process, which leaves more computational power for these processes. The haptic rendering system implemented is evaluated by haptically rendering a virtual prototype of a future cellular phone. The process for creating such a virtual prototype starts by designing a CAD model. The model is then converted to Open Inventor format which is the format used with haptic and visual rendering. The functionality and haptic properties are added to the converted model with a special tool to produce a haptically renderable functional virtual prototype. Experiences showed that adding functionality and haptic properties to static Open Inventor models and modifying these models is quick and easy. The realism of the sensations received from haptic rendering may be further developed by implementing a virtual fingertip which allows simulation of various fingertypes.

PREFACE

This study was carried out during 1997 as a part of the Virtual Reality Prototyping project (VRP) at the Technical Research Centre of Finland (VTT).

I would like to express my gratitude to the supervisors of this work, professor Petri Pulli and associate professor Kari Kuutti from the University of Oulu for their valuable comments to this study. I am also deeply grateful to the advisor of this work, Marko Salmela M. for his guidance and contribution during the study.

I would also like to thank Mikko Kerttula and Harri Kyllönen from VTT Electronics for their assistance.

Finally, thank you Titta for your support.

Oulu 1.2.1998

Tommi Anttila

CONTENTS

LIST OF SYMBOLS

3D	Three dimensional
Armlib	A haptic library
C++	An object oriented programming language
CAD	Computer Aided Design
DC	Direct Current
Ghost	A haptic library
GSM	Global Systems for Mobile Communications, a mobile phone system
haptic	To touch or having to do with the sense of touch; tactile.
HF	Haptic Feedback
HMD	Head Mounted Display
HW	Hardware
I-Collide	A collision detection library
I/O	Input / Output
Java	An object oriented programming language
Open GL	A software interface to graphics hardware
Open Inventor	3D graphics library
PC	Personal computer
SGI	Silicon Graphics Inc.
SW	Software
TCP	Transmission Control Protocol
UDP	User Data Protocol
VAS	Virtual Acoustics Simulator
VR	Virtual Reality
VRML	Virtual Reality Modelling Language
VRP	Virtual Reality Prototyping
VTT	Valtion Teknillinen Tutkimuskeskus (Technical Research Centre of Finland)

1. INTRODUCTION

There is a need to compress product development cycles and reduce costs when developing new products. Especially in the area of electronics that incorporate embedded computer systems, increased product complexity and global competition together with shortened product life cycles have given rise to exploration of new approaches. Virtual prototyping tries to overcome set challenges by providing a fully digital model of the simulated device. It uses simulatable computer models of a product which are referred as virtual prototypes [1].

The virtual prototype simulates a product's visual appearance, sound properties the functionality of the user interface and its behaviour as closely as possible. Virtual prototyping reduces the need for costly physical prototypes by providing product testing and evaluation without or in combination with physical models. The properties of virtual prototypes can be easily changed which makes it possible to quickly test many different design alternatives for the final product. A further advantage is that re-evaluation of the created virtual prototype is quick, what allows the designer to test more prototypes than would be with conventional physical prototypes. When the first physical model is produced, the need for changes should be minimal and the produced physical model should be very near to the final product. Virtual prototyping improves the accuracy and quality of the prototype to meet the needs of the customer and the market [1].

Conventional virtual prototyping, however, lacks the simulation of the physical properties of a real product in user-to-product interactions. Visual 3D models of virtual prototypes do not permit touch interaction, which is one of the most intuitive and expressive means for humans to interact with their environment. In our daily lives, touch interaction is used in many ways to interact with our environment. We explore objects by touching them and receive large amounts of information about their shape and materials, without paying special attention to it [2]. Visual virtual prototypes do not reveal the physical properties of the product or design errors that are connected to the physical properties of the product. Physical prototypes in turn show what the product looks and feels like, but do not usually provide the functionality of the end-product.

Haptic rendering [3] closes the gap between physical prototypes and visual virtual prototypes by giving the user the ability to sense the physical properties of virtual prototypes. It gives the user an opportunity to receive force and tactile feedback from 3D images of virtual prototypes and enhances that way the realism of virtual prototypes. Usually feedback from computer models is limited to visual and sound feedback, and manipulation of the models is performed with the keyboard and mouse. A haptic device is used for tracking the user's position and actively exerting forces on the user. It is the only interface that provides a two-way information flow between the user and a computer. With haptic rendering, testing of the virtual prototype's physical parameters such as material properties, button properties (size, shape, material, spring constant) is possible. Haptic rendering provides a realistic and comprehensive simulation of the physical user interfaces of products. It is much more effective for understanding if one can touch and use the virtual prototype rather than just see it and manipulate it with a mouse. A stereographic view with haptic rendering should provide the user the possibility of thinking that the rendered virtual prototype is as realistic as the physical model.

Haptic rendering increases the testability of virtual prototypes. As it is possible to change different material parameters, button parameters and the size of different objects, it is quicker and therefore easier to achieve good product designs without or together with real physical models. Without haptics, a significant amount of information is not available about the physical characteristics of the product. Haptic rendering of a virtual prototype provides a more complete view of the final product than conventional virtual prototyping based merely on visual techniques. Beside product development, haptic rendering can also be applied to many other fields such as medical applications, nanomanipulation, telerobotics, military applications and entertainment [3, 4].

Figure 1 shows different forms of human-computer interaction with virtual prototypes. It also shows examples of how different simulation models can simulate different properties of a virtual prototype.

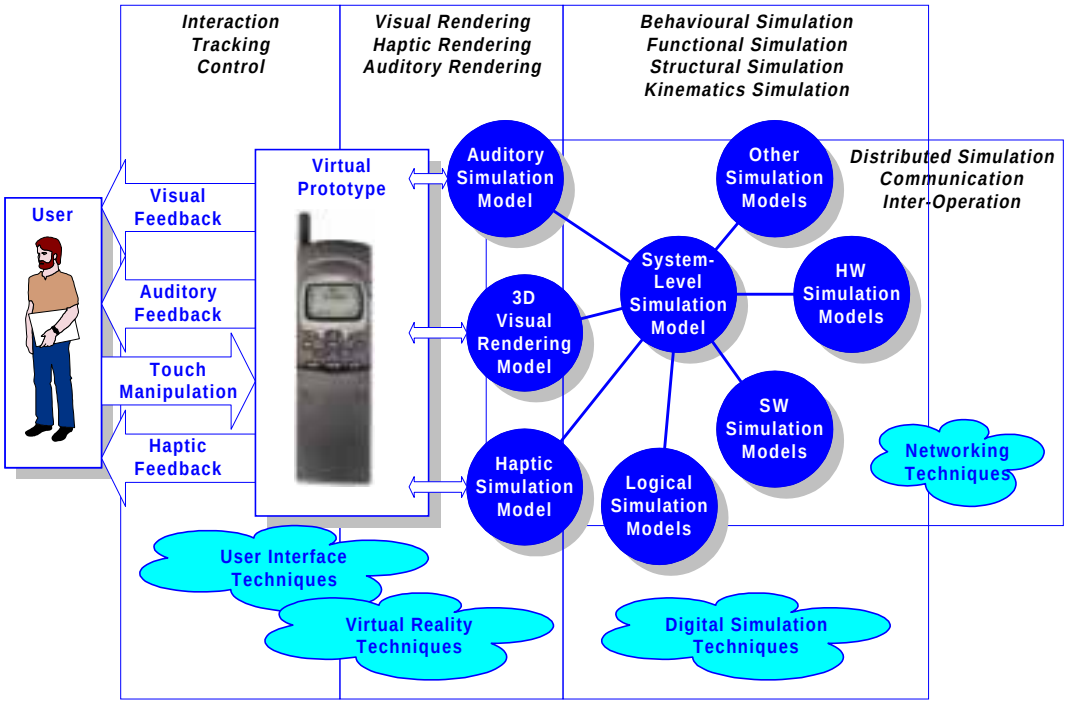

Figure 1. Human-computer interaction with virtual prototypes.

This work was carried out at VTT Electronics in the Virtual Reality Prototyping project (VRP) [5] and it was a part of research and development work on the VRP environment. The VRP project concentrates on developing virtual prototyping technology for product development in the area of electronics and telecommunications. The VRP project uses new virtual reality techniques to increase testability of virtual prototypes. The goal of this work was to create a haptic rendering system that provides haptic rendering of complex-shaped virtual prototypes of small, usually handheld, electronic and telecommunication devices. A very important aim was also to synchronise visual and haptic simulations so that no discernible delays between the visual and haptic environments would occur.

The objective of the haptic rendering system is to provide as realistic as possible sensations of the simulated prototype. To simulate the physical properties of an electronic product, several features should be haptically renderable (Figure 2):

- shapes
- buttons
- surface stiffness
- surface friction
- movable parts (sliders etc.).

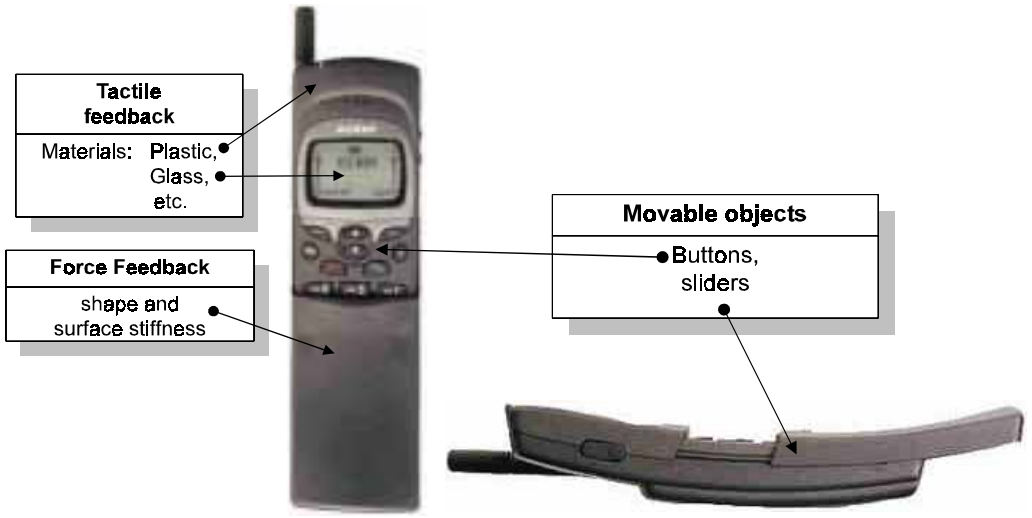

Figure 2. Haptically renderable properties of a cellular phone.

To enable the haptic rendering of the physical shape and the user interface of electronic and telecommunication products, the following tasks were set to:

- **Define the architecture and the components for a haptic rendering system** that would provide a stable and accurate environment for the haptic rendering of complex-shaped virtual prototypes.

- **Implement the haptic rendering system** utilizing selected software and hardware components.

- **Develop interfaces and modelling process support.** The implemented haptic rendering system should provide easy generation and modification of virtual prototypes with haptic rendering capabilities. The system should take into account special features that are related to the physical properties of small handheld electronic products.

- **Evaluate the implemented haptic rendering system** by testing performance and rendering properties.

This study concentrates first on describing haptic rendering and the software and hardware components that are required for a haptic rendering system. The constructive part of this work concentrates on finding and implementing an appropriate architecture for a haptic rendering system, and developing the selected architecture to be feasible for usability tests and development of new electronic products.

The haptic rendering system implemented is evaluated by haptically rendering a virtual prototype of a future cellular phone. The process for creating and modifying such a virtual prototype is described. Also the problems and experiences from haptic rendering in the implemented system are described.

2. DESIGN OF A HAPTIC RENDERING SYSTEM

The VRP project (Virtual Reality Prototyping) at VTT Electronics is aimed at developing virtual prototyping technology for product development in the area of *electronics and telecommunication.* Virtual reality techniques are used for providing a realistic rendering of the simulated product features and a user-interface to interact with virtual prototypes [5]. Concentration on small handheld electronic products allows adequate haptic rendering with present day high-end haptic devices.

A research environment has been constructed to study virtual prototyping (Figure 3). A virtual prototyping environment can consist of several computers each dedicated to a particular simulation area. Communication between different simulations is a socket-based [6].

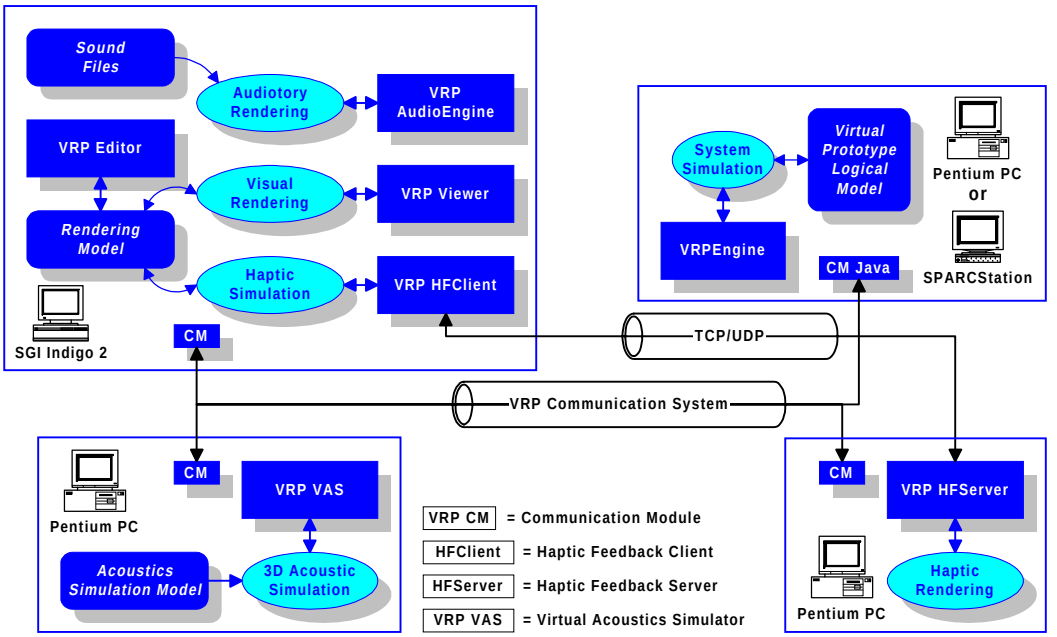

Figure 3. Schematic representation of the virtual prototyping environment at VTT Electronics.

The VRP project aims to develop a virtual prototyping environment with basic functions. An integration framework connects different simulations and existing software and hardware to virtual prototypes. A component management system quickens and eases the development of virtual prototypes by providing reusable virtual prototype components. Haptic rendering system requires special equipment to render virtual prototypes (Figure 4).

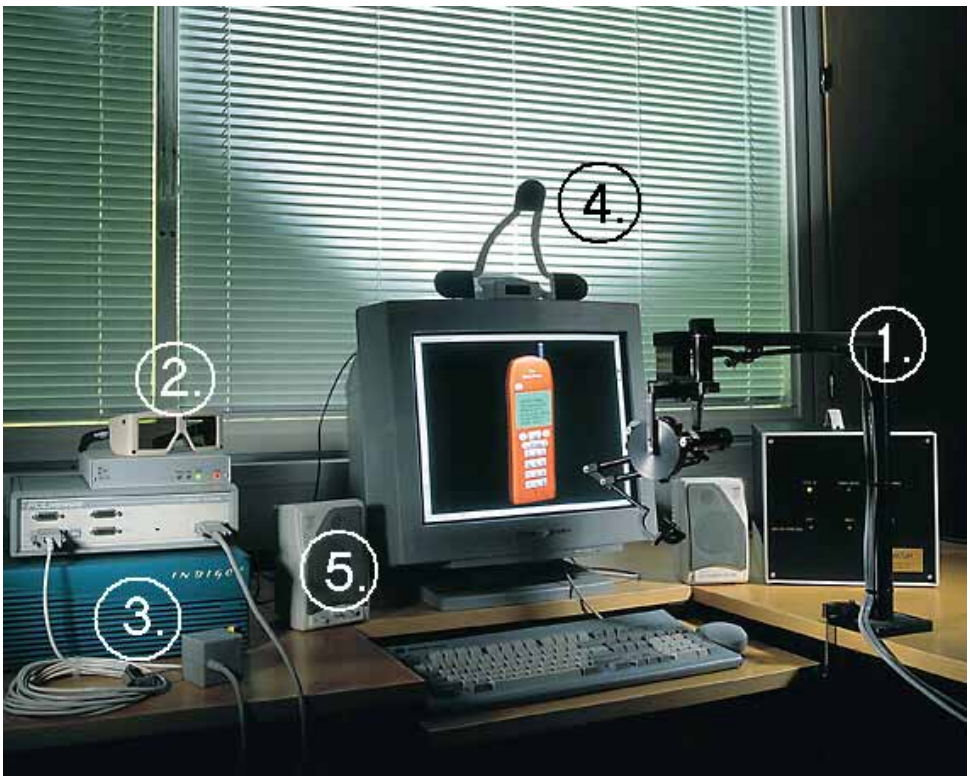

Figure 4. Virtual prototyping environment at VTT Electronics. 1) A haptic device, for receiving tactile and force feedback from virtual prototypes and for manipulating them by touch. 2) 3D glasses, to get a stereoscopic image of the simulated virtual prototype. 3) Efficient workstation, for haptic and visual rendering. 4) A head position tracker to find out the position of the head. When the user moves his/her head to look at the virtual prototype from the side, the visual scene changes so that it is possible to look at the object from different directions. 5) A soundcard with loudspeakers to output sound effects from the virtual prototypes.

2.1 Principles of haptic rendering

In the VRP project, haptic rendering means continuous measuring of the user's fingertip position and applying appropriate forces to user's index finger with the the used haptic device. The haptic rendering loop must be executed at a frequency of approximately 1000Hz to be able to give credible sensations of the simulated prototype (Figure 5) [7]. At slower update frequencies, virtual walls may feel unstable or soft, which decreases the level of realism.

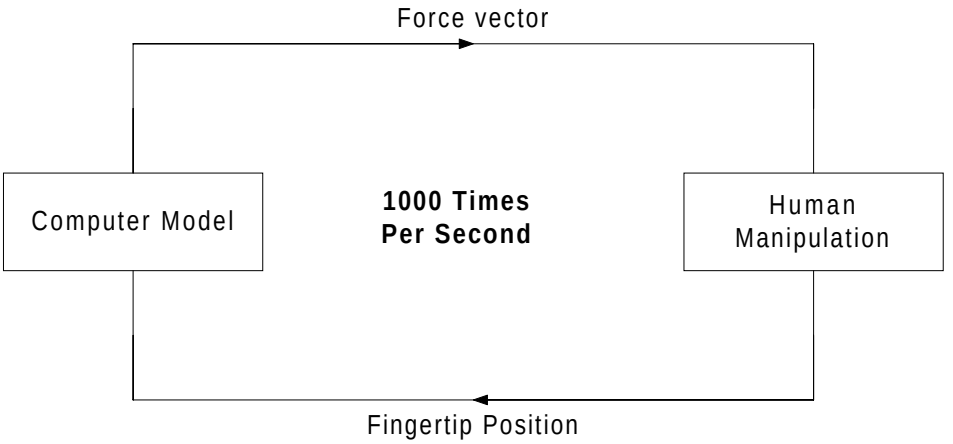

Figure 5. Haptic rendering loop.

The way in which a force vector is applied, can be based on the plane and probe model [8]. The plane and probe model applies forces to the probe (and the user's fingertip) depending on the probe's relative position to virtual planes (Figure 6) [4].

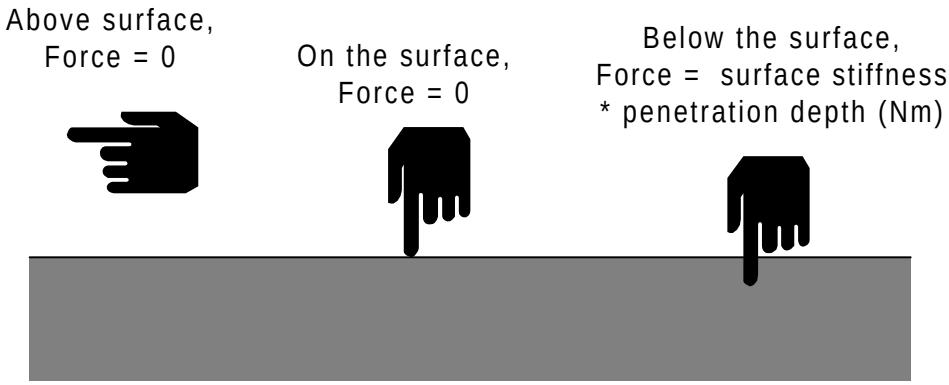

Figure 6. Applying forces in the plane and probe model.

Virtual planes are defined by local approximations of the actual surface. In this method, as accurate as possible local approximation should be maintained to get a good sensation of varying virtual surface [8]. An example of local approximations, when the user moves his finger in the direction of the arrow, is presented in Figure 7.

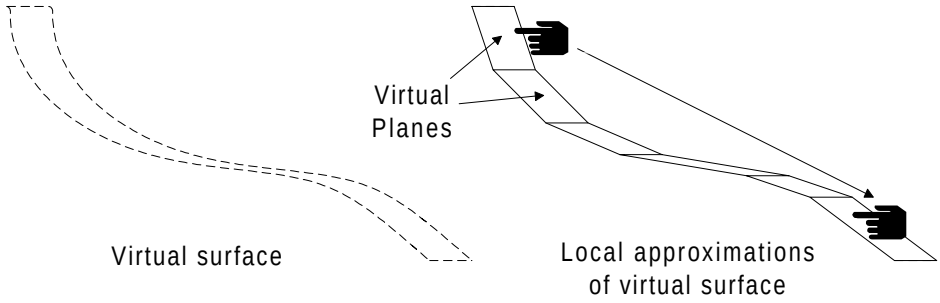

Figure 7. Local approximation method.

When the user moves the probe on a virtual plane, the plane equation is updated frequently to create a sensation of shape. When the correct virtual plane that the user is touching has been found, a resisting spring force towards the probe is defined by the penetration depth. As a result, the user can feel a virtual wall whose stiffness depends on the used parameters and properties of the haptic device [4].

Some applications need a different approach from virtual planes to achieve realistic haptic rendering. For example, moving objects that are under the influence of complex force-fields require a virtual spring model. A virtual spring pulls the user's finger towards the point of contact. Multiple virtual springs can together create a torque-effect [4].

2.2 Requirements of haptic rendering

Virtual prototypes of electronic products can be very complex-shaped. The ability to render these prototypes stable and accurate, sets high performance requirements on the computer used. The haptic rendering environment must be efficient enough to render complex virtual prototypes useful for virtual prototyping. Real-time visual and haptic rendering of realistic virtual prototypes is required.

Speed

The haptic rendering loop must be executed at a frequency of approximately 1000Hz to be able to give credible sensations of the simulated prototype. The visual update rate of 25Hz must be reached for smooth animation of visual scene. Haptic and visual scenes must be synchronised so that no perceivable delays occur when manipulating the haptic or visual environment. This is very important for the degree of realism that the rendering environment can provide.

Haptic properties

Haptic rendering should as closely as possible enable creation of the same physical properties as the actual physical product has. Rendering of static shapes includes all possible surface shapes such as shape primitives (cones, cubes, cylinders and spheres) and more complex shapes constructed from separate planes. With these it is possible to create any kind of static virtual prototype.

To achieve realistic sensations for the simulated product, haptic rendering of material properties is essential. A static virtual prototype without any surface friction makes the simulated prototype material feel like oiled glass [8]. With

material parameters, it is possible to simulate sensations of plastic, rubber and other materials. Simulation of surface properties contains haptic rendering of

- static friction
- dynamic friction
- surface stiffness.

Surface stiffness defines the hardness of the surface. Generation of hard surfaces requires special attention to the safety and stability of the rendering environment. A haptic device has the power to destroy or damage itself and hurt the user when faults occur [8]. In these situations, hard surfaces may increase the generated power that the haptic device applies.

Movable objects

Virtual prototypes must also contain movable objects. This means mapping of the model shapes to buttons, sliders and dials. Movable objects must be synchronised between the visual and haptic environment so that no perceivable delays occur when manipulating dynamic objects. When the user manipulates for example a button in the haptic virtual prototype, also the visual prototype changes synchronously. Then if the button is pushed to some predefined depth, event can be sent to the appropriate event handler for example to update the simulated product's display.

Usability

Physical attributes must be precise and easily changeable. When modelling a virtual button for example, changing slightly the spring force of the button may give a better feel to the product user-interface. After iterating the appropriate spring constant, it should be easy to construct a similar physical model. Selecting materials for different parts of the final product should also be easier after testing different materials with a virtual prototype.

To make more informative usability tests with small handheld electronic devices, some kind of approximation of the user's fingertip with haptic rendering is required. This would allow, for example, tests on if a button is too small to be pressed, or if buttons are too close to each other to be easily used.

2.3 The virtual fingertip

Haptic applications usually present the user's haptic fingertip as a small point, which can go through or get stuck in the tiniest holes in a 3D model [8, 9]. The visual fingertip is usually shown to be bigger than the haptical virtual fingertip for visibility reasons. This causes various problems in haptic rendering. When the user's fingertip is haptically presented as a small point, stable haptic rendering of a virtual prototype is harder. The user's finger is allowed to go inside little features, which in turn sets greater speed requirements on the haptic rendering to collision detection and back loop. Also, in cases when the size difference between the haptic and visual fingertip presentations is big, the user tends to make estimation errors.

A virtual fingertip [7] is an approximation of the user's physical fingertip. It presents the user's haptic and visual fingertips as equal-sized. For simplicity reasons, the virtual fingertip can be presented as a sphere, which quite well matches the properties of a human fingertip. The size of the sphere should be easily changed to simulate different finger types. This is very useful when making product usability tests.

Figures 8-13 below describe the most common problems and incorrect behaviour that occur when the haptic and visual fingertips are different sizes. Figures 8-13 a) describe the situation where the haptic fingertip is a small point and the visual fingertip is a larger sphere. Figures 8-13 b) describe same situations when the haptic and visual fingertips are equal in size.

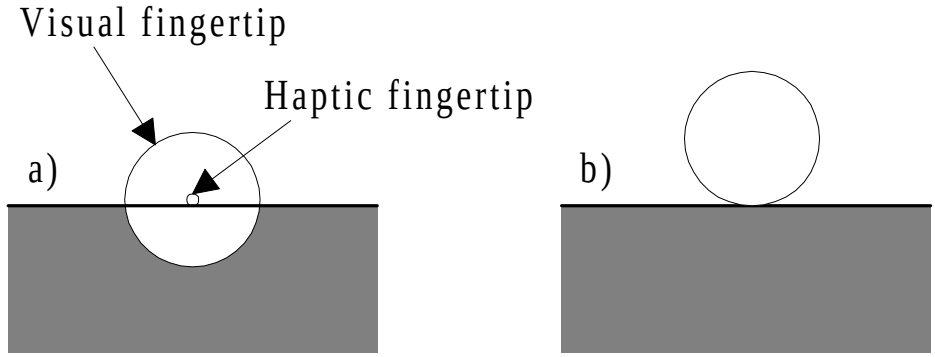

Figure 8. User's finger on a virtual surface. Figure 8a) shows how half of the virtual fingertip is allowed to be and slide inside a virtual plane. This does not look very realistic, showing a solid looking object is not solid after all. Figure 8b) shows the correct functioning.

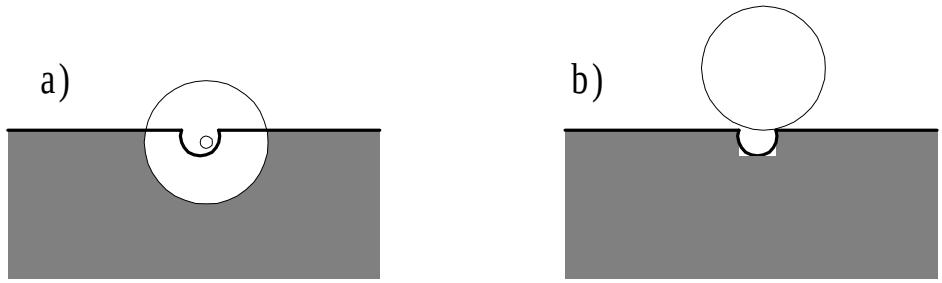

Figure 9. Small hole on a virtual surface. Figure 9a) shows a case, where the user gets stuck in a small hole on the surface. The virtual fingertip should not be able to get stuck in small holes as shown in 9b) because the rendering of these shapes is very power consuming. It has also an effect on the stability of the haptic rendering system. In most cases, it is better to close these small holes to improve the quality of haptic rendering.

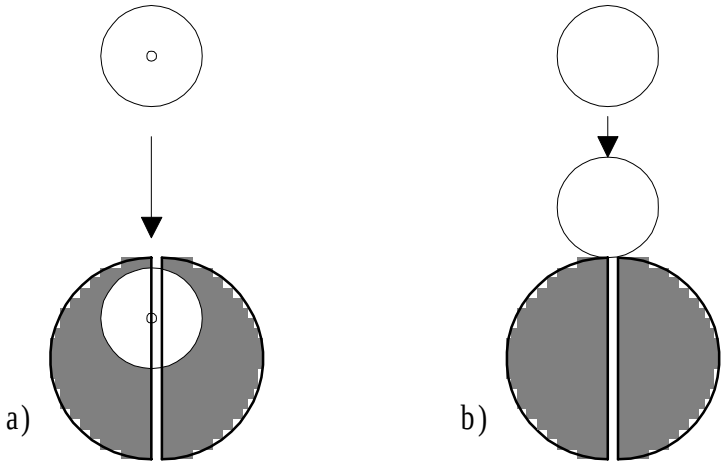

Figure 10. A narrow gap between virtual prototype's parts. Figure 10a) shows how the user's fingertip can slip inside and through a virtual prototype from a small gap between parts when the user thinks that the haptic and visual fingertips are of equal size, unlike in 10b) where the user's finger is not allowed to go through the prototype. Because of this, special care must be taken when assembling the virtual prototypes from the virtual objects. Adjacent parts in a virtual prototype must be tightly joined to close these small holes.

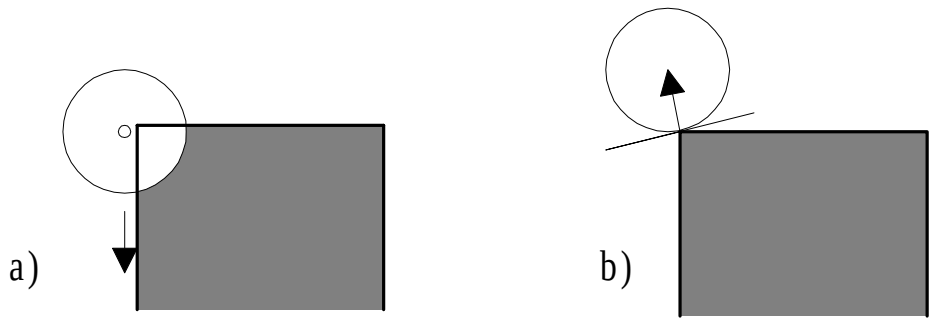

Figure 11. Pushing a button. Figure 11a) shows a case where the user sees a bigger finger-object than it haptically actually is and gets a false feeling that the fingertip is stably pushing a button. This is not however the case. Often this false feeling of stability causes the user to slip from a button. In Figure 11b), the haptic fingertip is the size as the visual and problems do not occur as easily.

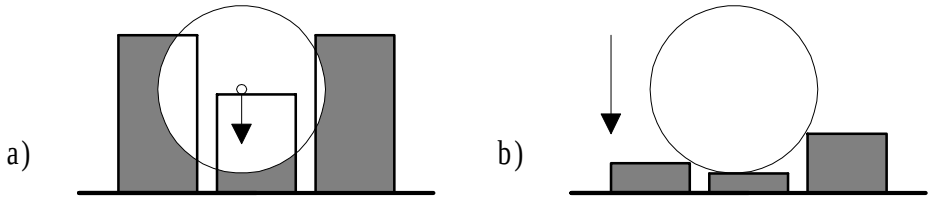

Figure 12. Pushing buttons that are near each other. Figure 12 shows a case where the user pushes the buttons that are near each other. In Figure 12a) the user is allowed to be partially inside other buttons. Some problems are quite similar to those described in Figures 8, 9 and 10 but also another problem occurs. The user is not able to push the multiple buttons at the same time as in the real world. Figure 12b) shows the correct functioning in this situation.

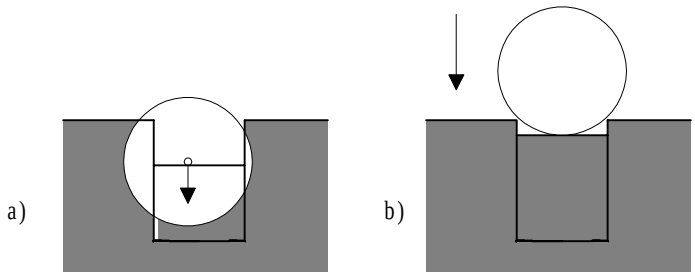

Figure 13. Pushing a button that is in a hole. Figure 13a) shows how the user is able to push the button deeper than should be possible. Figure 13b) shows what would happen in the real world: the button is too deep to be pushed for a finger of this size or the button is too small.

The virtual fingertip increases the stability of the haptic rendering by avoiding incorrect haptic rendering of small details as shown above. It also increases the realism and improves the amount and quality of the information from virtual prototype usability tests. It gives for example information as to whether the buttons are too small to be pushed, if the buttons are too close each other or too small.

2.4 The Phantom haptic device

The haptic device is the basic component in haptic rendering. It defines the haptic interface between the user and computer: what kind of sensations the user can receive. The selected haptic device for the VRP project, Phantom, uses a passive thimble as a user interface [10]. The mechanics of the Phantom are shown in Figure 14 [11]. The system consists also of a separate power amplifier cabinet and an I/O card that connects the Phantom to a computer that controls it. With a single Phantom haptic device, tactile and force feedback can be received with one finger.

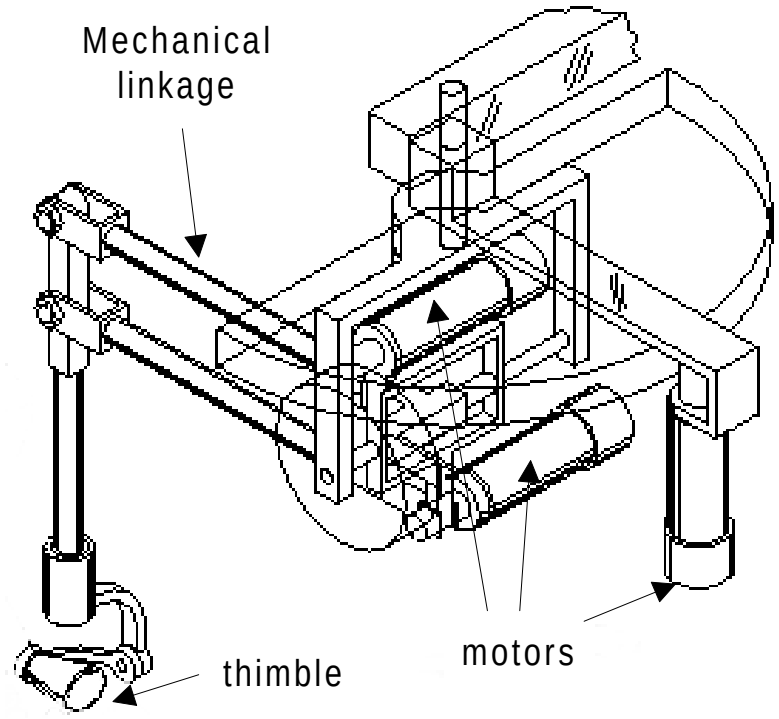

Figure 14. The Phantom haptic device.

The Phantom uses a small wrist centred workspace which is adequate for use with virtual prototyping of small handheld electronic devices. A passive thimble as a user interface means that the rotation angles of the users fingertip are not

measured but it is only possible to measure the position. There is, however, an option for an encoder gimbal for the Phantom to measure rotation angles [11].

Forces are sent to the user's fingertip via a lightweight aluminium linkage with three DC brushed motors. The position is determined by 3 optical encoders mounted on motors. The peak maximum force that the Phantom can produce is 8.5N [11] and the maximum continuous force is about 1.5N [10]. These values should be sufficient to produce realistic sensations of virtual prototypes with hard surfaces when reasonable forces are used by the user. At the design phase of the Phantom, the main focus has been on achieving low mass, low friction, low backlash, high stiffness and good back-driveability [10]. These properties enable accurate rendering of shapes and frictions.

A single thimble as a user interface decreases the realism of the sensation that can be achieved. The most natural interface would include all the fingers in haptic rendering. It is however possible to use two Phantom haptic devices with the thumb and index-fingers at the same time to achieve a grasp-effect as shown in Figure 15 [11].

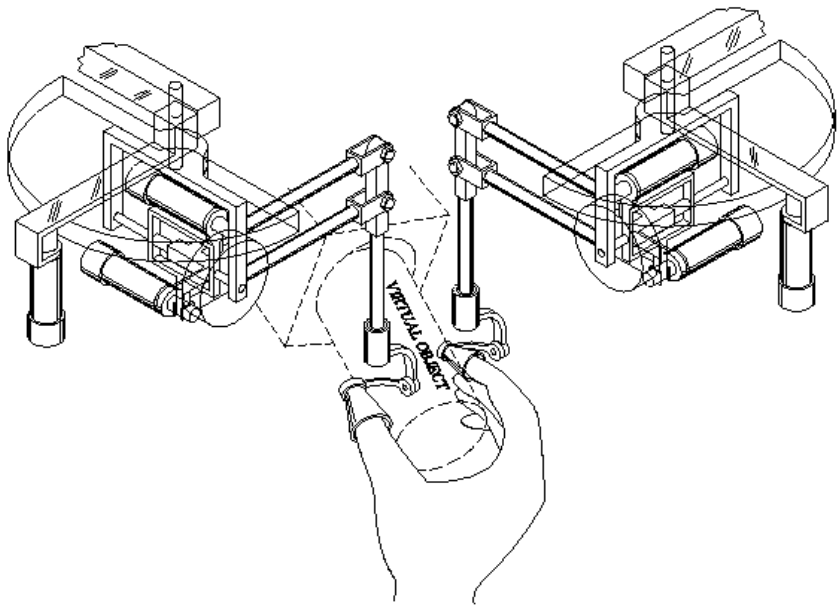

Figure 15. Grasp effect with two Phantoms.

The Phantom can be controlled with PC computers and SGI workstations. There is also a super extended workspace Phantom with larger maximum forces and workspace available for rendering larger virtual prototypes [11]. Other commercial haptic devices with same cost/quality ratio are not really available. Phantom specifications [11] are described in Table 1:

Table 1. Phantom specifications.

Property	*Standard Workspace Phantom System*	*Expanded Workspace Phantom System*	*Super Expanded Workspace Phantom System*
Nominal position resolution	0.03mm	0.03mm	0.02mm
Workspace	13x18x25 cm	19.5x27x37,5 cm	42x59x82 cm
Backdrive friction	0.04 N	0.04 N	0.02 N
Peak Force	8.5 N	8.5 N	22N
Closed loop stiffness	3.5 N/mm	3.5N/mm	1 N/mm
Inertia	< 75g	< 75g	< 150g

2.5 Constraints on the haptic rendering system

Constraints on a haptic rendering system are set by the properties of the haptic device used, the software and hardware components, and the processing power of the computers that are used for haptic rendering. As described before, a haptic rendering loop must be executed at least at the frequency of 1000Hz and the required processing power increases when the complexity of the rendered virtual prototype increases or the size of the details decreases.

The haptic device used defines the maximum stiffness of the virtual surface. The peak maximum force that the Phantom can produce and the maximum continuous force are quite close to those of human capabilities. The maximum force of the human finger is 40N but people rarely exert more than 10N. In

normal usage, the average force that the human finger produces is about 1N [10].

In the VRP project, the haptic and visual environments are not overlapping for several reasons. First, the technique used for stereographic rendering is not able to bring the 3D image far from monitor screen so that user would be able to see the image clearly without strict concentration. The 3D image has to be brought far from the monitor screen for safety reasons, so that the Phantom cannot reach the monitor screen and break it in any situations. A head mounted display (HMD) would visually enable bringing visual and haptic spaces together. Because the basic Phantom does not have the ability to measure rotation angles and the fact that the contact point is not at the fingertip but it is rather at the midpoint of the user's finger where the rotation of the finger does not effect the position of the visual user's fingertip, this is not used. When the user's manipulation finger is in contact with a virtual surface, physically it would be partly inside the virtual prototype and when the user rotates his/her finger at the surface of the object there would be no effect. For this reason, visual and haptic prototypes are apart as shown in Figure 16.

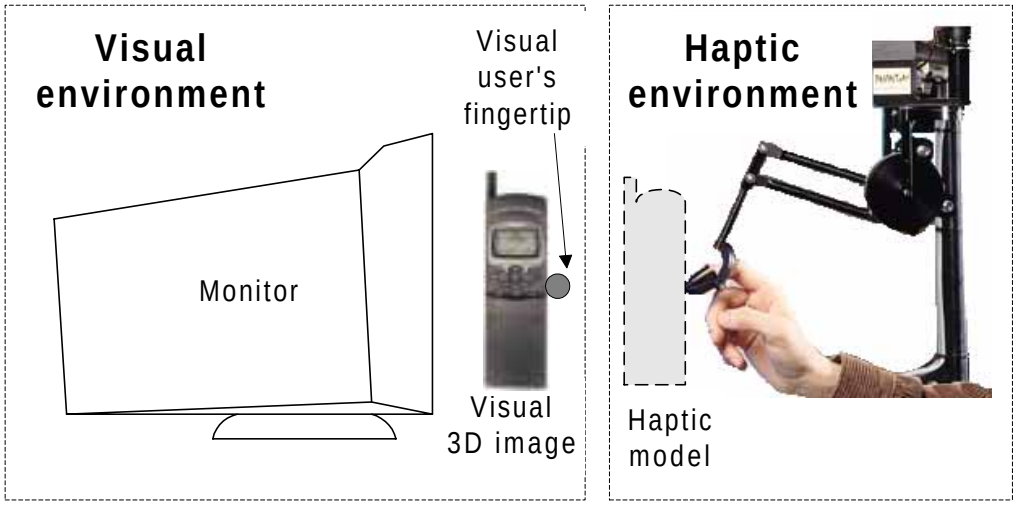

Figure 16. Haptic and visual spaces separated.

The Phantom limits the touching capability to one finger with one Phantom haptic device. This reduces usage possibilities and the realism of the haptic rendering.

Versatility of the surface friction parameters defines the capability to create credible material simulations. When there are multiple parameters that have an effect on the simulated surface, setting the appropriate values is more difficult but there should also be a possibility to achieve more realistic material sensations. Also the capabilities of the haptic device used effects the realism of the created sensations of materials.

2.6 Architecture design

The architecture of a haptic rendering system plays an important role in the performance of haptic rendering. An underperforming hardware architecture causes communication delays and therefore decreases the maximum complexity of a virtual prototype. It also effects the accuracy and stability of the haptic rendering. Software architecture defines which tactile and force feedback is possible to simulate and how realistic it is. The basic processes needed for haptic rendering in virtual prototyping are shown in figure 17.

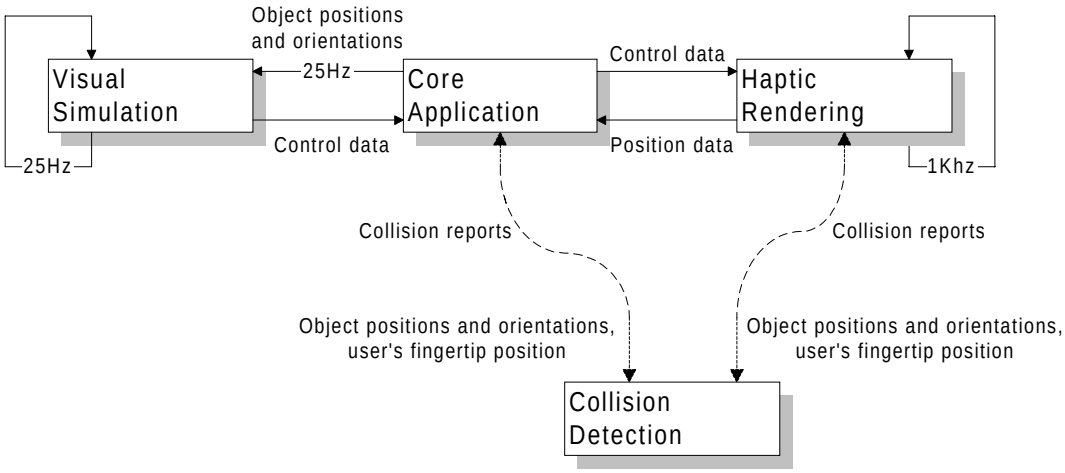

Figure 17. Basic processes of haptic rendering.

Core application is used to initalize and control other processes that are included in the haptic rendering system. After initialization, the core application's task is to keep haptic and visual environments synchronized so that no delays between haptic and visual environments occur. In some cases, the core application also controls *collision detection.*

As can be seen, the communication rate between the *visual simulation* process and *core application* is relatively slow that it is possible to separate the visual simulation process into another computer from other processes. The core application's task is then to send position and orientation data to the visual simulation process to synchronise haptic and visual environments. Separating the visual simulation process into a different computer from other processes may considerably increase the performance of the haptic rendering.

Collision detection may be a separate process or it can be a part of the *haptic rendering* process. If it is not a part of the haptic rendering, a decision about where to run this process must be made. Collision detection may be a heavy task when the haptic rendering is applied for a complex 3D visual model.

Separating collision detection off to a different computer from the haptic rendering process increases the communication overhead which in turn can introduce delays and lead to problems in the quality of the haptic rendering. The frequency with which the loop, from measuring the user's fingertip position to collision detection and sending forces to user's fingertip, is executed defines how small details can be haptically rendered. If this loop is not fast enough, small details remain unrendered or are rendered incorrectly. Incorrect rendering happens when the user's finger moves too fast so that correct virtual surfaces cannot be determined in time. In these cases, the user's fingertip is able to go inside or through a virtual object. Separation of these tasks into different computers however leaves more computational power for these tasks, which in some cases may lead to a better result.

The *core application* is not necessarily itself a heavy process, which means that it can be executed with, or encapsulated to, one of the other processes. In the case where collision detection is not a part of the haptic rendering process, the *core application* and *collision detection* processes may be combined.

Haptic rendering requires a lot of computational power. The speed at which the haptic process loop is executed defines the stability of the stiff virtual walls. It may be reasonable to separate it off to a different computer, at least from the visual process.

3. IMPLEMENTATION OF THE HAPTIC RENDERING SYSTEM

The task of haptic rendering consumes very much computer power. One of the main issues in virtual prototyping is the ability to render complex shaped haptic prototypes. This ability depends not only on the available processor power but also on the selected architecture.

The software architecture defines the haptic properties that can be rendered in a haptic rendering environment. The hardware architecture divides the selected software components to different computers and therefore defines the available processing power for different processes and communication delays between those processes.

3.1 Software components

A haptic rendering system should have the following basic software components

- a haptic renderer
- a 3D graphics renderer
- a collision detection algorithm
- a communication system.

A haptic renderer creates haptic illusions of 3D objects. It is used for calculating the position of user's fingertip and the force and tactile feedback that is generated for the user through the haptic device used. The haptic renderer also defines what kind of haptic properties can be simulated.

The purpose of a collision detection algorithm is to give information on all collisions between virtual objects. Based on this information the object’s motion should be constrained so that objects should not be able to pass through each other. A lack of efficiency in the collision detection algorithm may be a bottleneck in the haptic rendering of complex virtual prototypes [12]. The requirements of the collision detection algorithms may change. For example,

some algorithms demand that objects must have closed volume or objects must be convex.

A 3D graphics renderer is not essential in haptic rendering but it is very important in virtual prototyping. It is also recommended that a stereoscopic viewing system is used because of the increased reality of the simulated interaction with a virtual prototype and better matching of sensoral inputs.

The communication system plays an important role at the distributed approach of haptic rendering system. Depending on how the decoupling has been implemented, communication may require very fast throughput times.

3.1.1 Open Inventor

Open Inventor is an object-oriented 3D graphic toolkit based on OpenGL [13]. It is a library of objects and methods used to create interactive graphic applications. The basic idea is to create 3D objects from which all information is stored in a scene database. An Open Inventor scene is based on a hierarchical tree of nodes. Nodes represent geometries, materials, translations and behaviour of 3D objects. Nodes can be grouped and then effects can be applied to these subtrees: to visually render the scene from a node tree, or apply user defined methods to the node tree.

Visual simulation in a Silicon Graphics workstation with Open Inventor offers different viewers and stereographic rendering of 3D objects. Usage of stereographic rendering gives a more realistic view of the simulated world as it seems that the simulated prototype is floating in front of the monitor screen. Figure 18 shows an example viewer of Open Inventor. Thumbwheels are used for rotating, moving and scaling the object.

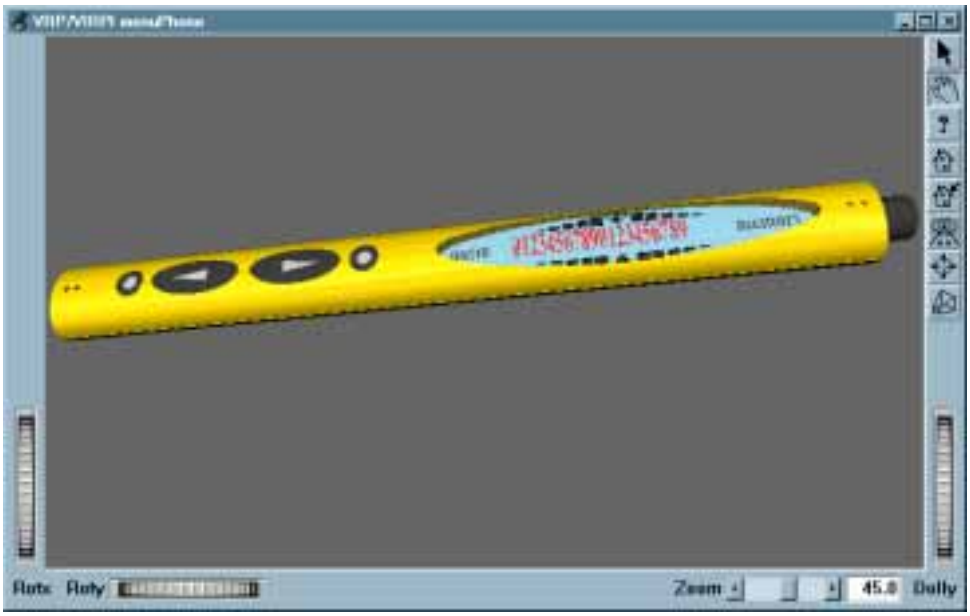

Figure 18. Scene of pen-shaped GSM phone with an Open Inventor viewer.

Open Inventor models can be made functional. This means that, for example, buttons on a virtual GSM phone can be clicked with a mouse, and the display of the simulated prototype is updated when a button has moved to a predefined depth.

A nodetree can be created with Open Inventor by loading an Open Inventor format file. Open Inventor also provides a good method for extending loading capabilities from the basic Open Inventor format. It is possible to create one's own classes and then parse these classes to the node tree [14]. This provides a good way of extending parsing methods of, for example, a haptic library.

3.1.2 I-Collide

I-Collide is a collision detection algorithm that uses a two level approach based on pruning multiple object pairs using bounding boxes and then performing exact collision detection between selected pairs [12]. At a lower level, the algorithm maintains a pair of closest features for each convex polytype pair, and calculates the distance between the features to detect collision. This is used to determine the contact status between polytypes. Only distances between user

selected polytype pairs are significant so that all pair's distances of all pairs do not have to be maintained. The pruning algorithm reduces the number of pairwise collision tests by eliminating distant polytype pairs. The method for finding closest feature pairs is based on Voronoi regions [15].

I-Collide has some major constraints that should be considered when selecting a collision detection algorithm for haptic rendering. Constraints of the I-Collide are

- temporal coherence
- convex polytypes
- closed-volume objects.

The temporal coherence requirement means that objects can not travel too large distances between successive collision tests. When the speed of objects is high, the time between collision tests should be very small. This should be considered when selecting architecture of the haptic rendering system. For example, fast movement of the fingertip in the haptic scene may result in incorrect information in collision tests which in turn may lead to too large forces. A similar effect may happen when the rendered details are so small that even a slow movement of the finger causes frequent updating of the virtual plane.

Major drawbacks in I-Collide are also the constraints of convex and closed-volume objects. Concave objects must be divided to convex ones which makes automatic loading of virtual prototypes from a file more complicated. It is also very difficult to fill small holes from objects constructed of thousands of virtual planes if this matter has not been considered at the time of design of the original prototype with CAD.

The usage of I-Collide requires first that all 3D objects have to be instantiated. Then the decision has to be made about which pairs of objects should be tracked for possible collisions. Then it is possible to update the object positions and make collision tests [16].

3.1.3 Armlib

Armlib is a tactile and force feedback library that provides a simple and efficient function interface to control the Phantom, the Sarcos Research Corporation Dextrous Master Arm, and the Argonne Remote Manipulator haptic devices [8]. Armlib provides functions to control, acquire position and other information from the Phantom. Forces are sent to the Phantom directly or via virtual planes and springs. Armlib also provides a facility to implement surface materials to virtual planes.

Virtual planes have been implemented in Armlib by using local approximations of surfaces. In this method, as accurate as possible local approximation should be maintained to get a good sensation of the varying virtual surface. When the user moves his/her finger on the virtual surface, the plane-equation should be updated frequently depending on the speed that the user moves his/her finger.

Armlib is based on a client-server architecture by separating haptic and application processes as shown in Figure 19. This should leave more processor power for visual and haptic processes as they are separated. It is, however, questionable if this increases or decreases the maximum complexity of the virtual prototype that can be rendered. As has been previously described, the loop from the haptic rendering (measure the position of user's fingertip) to the collision detection (find the plane that the user is colliding with) and back (send forces to the user's fingertip) defines, how small details can be rendered in a haptic rendering system. With complex virtual prototypes, the client-server architecture may increase communication delays by more than the time achieved by separating the processes to get more computational power.

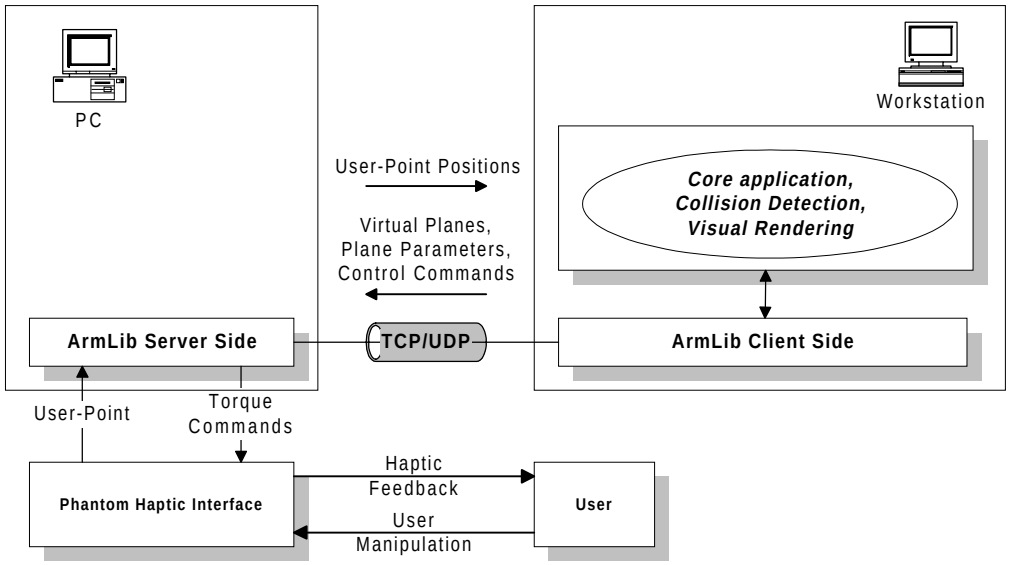

Figure 19. Armlib client-server system.

The basic communication protocol between the Armlib-server and client is TCP/UDP. The protocol does not guarantee that all packets will get through but when a secure message change is required, the function library internally implements secure message changing. The benefit from this protocol is that it is possible to achieve very fast communication lead-times because of the lack of acknowledgement messages [17].

The Armlib-server measures the position of the user's fingertip and sends it to the Armlib client depending on the selected position reading mode. The data reading mode is selected before reading the data. In asynchronous mode, the server sends the position of the Phantom gimbal as fast as possible to the client whether or not the client uses the data. In synchronous communication, the client sends a request to the server that processes the message and sends acknowledgement back to the client. In this case, the client must wait while the message makes a round trip and the server performs the desired action. All basic interactions between the client and the server are accomplished in this way. Reading the position may also be sent via synchronous communication, but this is not recommended.

After acquiring the position of the Phantom gimbal, the user application should decide what kind of effect should be applied to the gimbal. There are three basic ways to send forces to the Phantom [8]

- To send forces directly to the Phantom by giving the x-, y- and z-force in Newtons
- To send virtual planes to the Phantom by giving the normal to the plane and the stiffness of the surface in Newtons/meter
- To attach a virtual spring to the Phantom gimbal by giving the spring constant of the attached spring.

Armlib provides simulation of different surface properties. With Armlib, it is not only possible to change the stiffness of a plane but also the friction that the user feels when moving his/her finger on the surface of the object. The friction model is based on five different parameters which should enable the creation of realistic surface materials.

Armlib provides also a primitive way of implementing a virtual fingertip. It can be created from separate probe points which can be constrained by different virtual planes.

3.1.4 Ghost

Ghost [9] is an object-oriented 3D haptic toolkit used with a Phantom haptic device. It offers a C++ library of objects and methods that can be used to construct different haptic applications.

The rendering of a haptic scene has been made easy. Ghost provides a means of constructing a static haptic scene by reading it from a VRML 1.0 or Open Inventor format file, or by coding the scene directly to the application. When the scene is ready, Ghost automatically takes care of rendering the haptic scene after calling an appropriate method.

In Ghost, core application and haptic processes are executed on the same computer. Ghost does not provide or require any particular visual rendering method and does not therefore define where the visual process should be executed. Examples supplied with Ghost used OpenGL for visualisation and all

processes were executed on the same computer. Ghost contains a collision detection algorithm so that no separate collision detection algorithm is required or possible to use. This also eliminates the need for a separate plane selection method after collision detection as is required with Armlib.

The haptic rendering process gets as much processor time as it needs with Ghost. When not enough processor power is available for haptic rendering process, the application is killed for safety reasons. When there is processor time available from the haptic process, other processes can be executed. Callback methods can be used for receiving control from the haptic process when, for example, a button has been pushed and an event should be sent. Callback methods are also used for updating the visual scene at a sufficient rate.

The main haptic features that can be rendered with Ghost are

- shape primitives: cubes, spheres, cylinders and cones
- complex shapes constructed from separate planes (PolyMesh)
- static and dynamic friction
- buttons
- dials
- sliders
- haptic smoothing
- special effects: inertia, buzz and constraint.

A smoothing parameter defines how the edge between separate planes is smoothed. Figure 20 shows how a virtual angular surface can be smoothed to correspond a rounded surface.

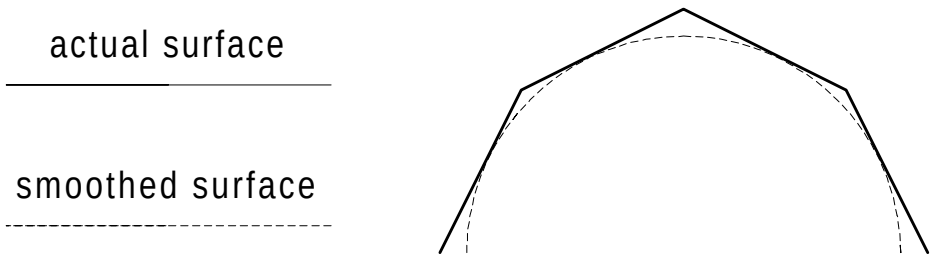

Figure 20. Smoothing an angular surface.

The haptic scene in Ghost is a hierarchical tree of nodes. Ghost does not provide the means to read extended dynamic Ghost nodes (buttons, dials, sliders, smoothing, frictions and special effects) from a file. This makes automatic creation of useful virtual prototypes more complicated. The Ghost haptic library does not support virtual fingertip, which may be the biggest drawback of this haptic library when used for usability testing of simulated physical user interfaces.

3.2 Architecture alternatives

Selecting the best software and hardware architecture is highly desirable when trying to simulate complex haptic virtual prototypes with realistic haptic sensations. Hardware architecture depends largely on the required computational power, whereas software architecture design always requires basic components for at least haptic and visual rendering. Also separate communication and collision detection software components may be required.

When selecting hardware architecture, questions arise about how many computers are required and what platforms should be selected. As the number of computers rises, computing power also rises. However, improper allocation of software components between computers may also increase communication overhead and delays remarkably. It is also more convinient to carry out simulation when there are only a few computers.

When the proper software components have been selected and an efficient architecture for software processes is used, it is possible to achieve accurate and stable haptic rendering of complex virtual prototypes.

3.2.1 Fully Distributed Architecture

The Armlib haptic library is a central component in this architecture approach. Other required components are Open Inventor, for visual rendering and I-Collide for collision detection. These three heavy processes are separated onto different computers so that they can get as much processing power as needed. The haptic process runs on a 100MHz Pentium PC dedicated to it, the visual rendering is

executed on a Silicon Graphics Indigo2 workstation and collision detection on a Sun workstation.

When using three computers, communication delays may become too long. Therefore the use of software components and the location of main application and virtual plane selection are the most important issues. Architecture design is shown in Figure 21.

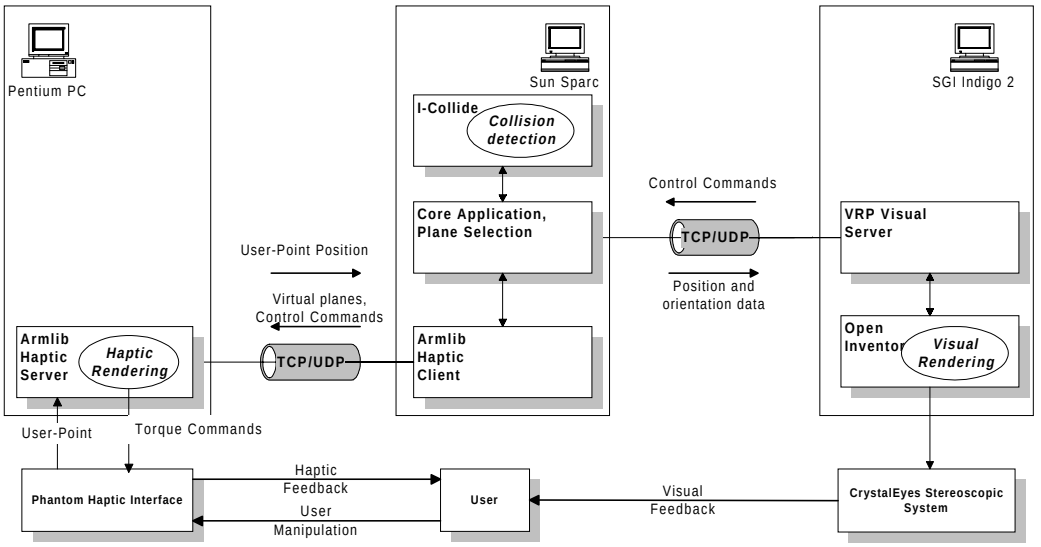

Figure 21. Fully distributed architecture.

Armlib divides application and haptic processes onto separate computers. This architecture moves also visual process to a separate computer. This architecture maximises the available processing power to processes. At the same time it does however also increase communication delays which effects the quality of the haptic rendering.

This architecture was not tested because of the need for three computers, which seemed too much for convinient use. It was decided that architectures with fewer computers should found.

3.2.2 Haptic Server Architecture

This approach uses a distributed architecture and the base component in it is the Armlib haptic library. In this architecture a 100MHz Pentium PC handled haptic rendering of the virtual prototype and a Silicon Graphics Indigo2 workstation handled core application, collision detection and visual rendering of virtual prototypes as shown in Figure 22.

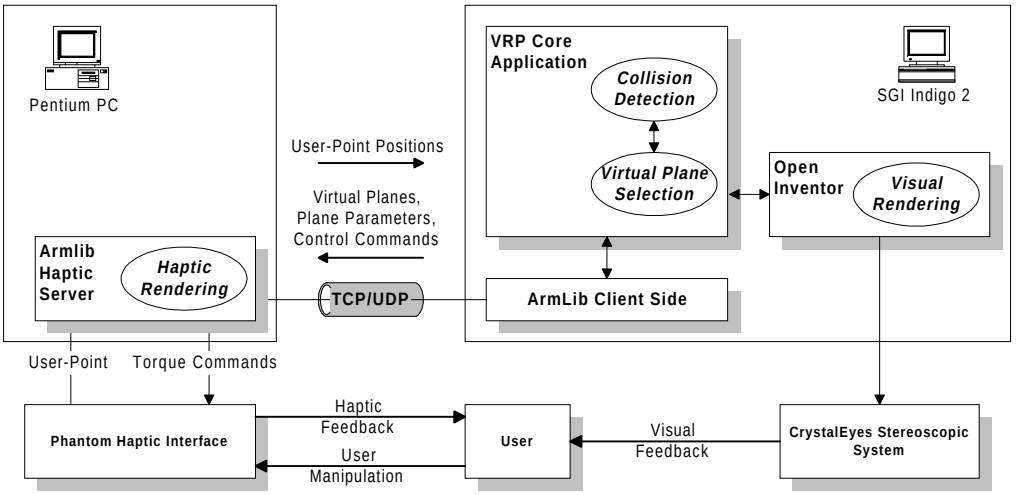

Figure 22. Haptic server architecture.

Haptic rendering is started by loading and parsing an extended Open Inventor format file in the viewer application. In addition to basic Open Inventor classes, extended Inventor database classes had to be created

- create haptic buttons
- include surface materials and frictions in the simulation
- register shape objects to collision detection data structure
- query I-Collide for results of the collision test
- update object positions in visual simulation.

Haptic buttons contain information about the pushing direction, spring constant of the button and the depth when button event should be sent. Surface material

nodes contain information about surface stiffness and parameters that together create sensations of static and kinetic friction.

After parsing the Open Inventor node tree, all shape nodes are registered to the I-Collide collision detection library. The registration method created registers the shape-object's planes one by one in the collision detection data structure. Next, the communication modes have to be initialised. Communication modes for reading the user's fingertip position and sending of virtual planes have to be initialised before starting the main application loop. Use of an asynchronous position read method increased considerably the throughput of the system compared to synchronous read.

The main loop of a core application process is described in Figure 23. The user's fingertip position is read as quickly as possible. Then the collision detection finds with which objects the user's fingertip is colliding. The actual collision planes have to be calculated to be able to send them in the correct form to the Armlib server. Before sending planes to Armlib server, previous useless planes have to be turned off and only the new planes that are not already activated are sent to the Armlib server.

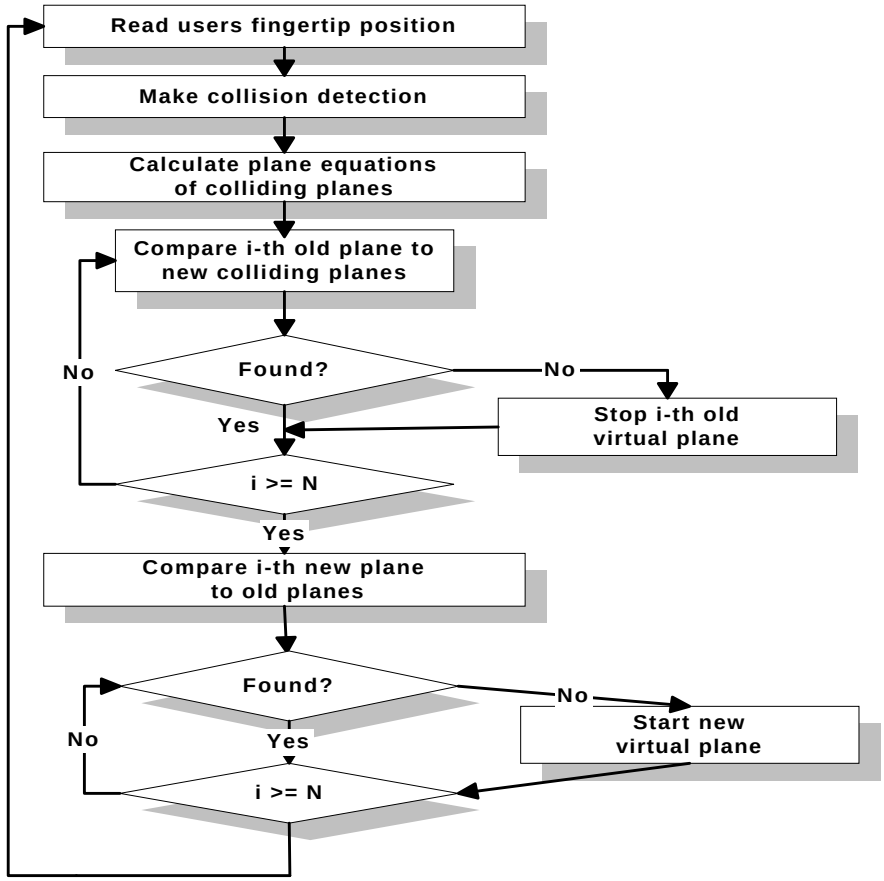

Figure 23. Main application process flow.

The main benefits of this architecture were that the software components were extendible and the distributed nature of the system allowed efficient rendering of simple haptic virtual worlds. The main weaknesses of the architecture were the constraints of closed volume objects, the lack of haptic shape primitives such as cubes and spheres, and the lack of haptic smoothing. The architecture used was efficient enough to simulate simple haptic worlds. The architecture was successfully tested with a virtual prototype of a simple mobile phone that included a cubic cover, 12 cubic buttons and a display, with surface materials. The maximum performance of the architecture was not tested but the rendering of complex virtual prototypes supposedly needs more computational power than this architecture with the computers used had. It is questionable if this

architecture allocation is efficient for complex virtual worlds. The communication delay between the haptic rendering process and collision detection with virtual plane selection may be too long.

3.2.3 Single Workstation Architecture

The central component in this architecture was the Ghost haptic software development toolkit. Ghost is used for haptic rendering and no separate collision detection algorithm is required. The only additional required component is for visual rendering. In this architecture all simulations are performed in one computer that takes care of the main application and the haptic and visual rendering.

Because all simulations are executed in one computer, clearly this computer should have a large amount of processor power. In the VRP project, the PC computer used was an Intergraph TZ425 that had two 266MHz Pentium II processors, 128Mb:s of operating memory and a powerful 3D graphics card (RealiZm Z13-T) for visual rendering [18].

Tests were made to see if all the simulations could be performed in one computer with a complex virtual prototype. The haptic rendering was handled by Ghost and the visual simulation was handled by a test program that used OpenGL. Tests showed that it was not feasible to do all the simulations in one PC. Haptic rendering of a complex virtual prototype without any dynamic behaviour could not maintain the stability and accuracy needed. The inability to use stereographic glasses with the Open Inventor on the PC platform was also a serious problem.

The benefits at this architecture are

- only one computer is needed
- the approach does not require any communication with other computers and therefore it also lacks communication delays
- separate collision detection is not needed.

These advantages compared to the previous architectures makes this architecture very attractive. The need for only one PC makes the architecture highly portable

and convinient. The drawback is that when simulating complex virtual prototypes, a very powerful computer is needed.

3.2.4 Visual Server Architecture

The main component at this architecture is the Ghost haptic software development tookit. To be able to simulate complex virtual prototypes, distributed architecture is used. The separation of the processes is shown in Figure 24.

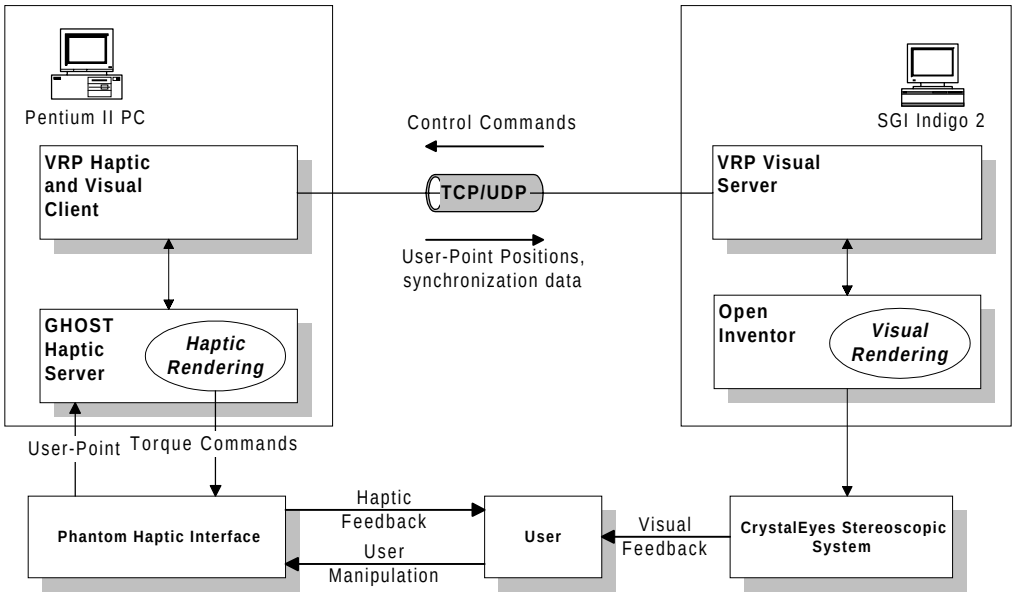

Figure 24. Visual server architecture.

The separation of the processes is based on the communication speed need and the characteristics of the computers used. The double-processor PC had more calculation power than the Indigo 2 workstation, whereas, the Indigo 2 had better visual rendering capabilities. TCP/UDP was used to minimise the communication delay from PC to SGI.

Communication between computers needs only 30Hz frequency from PC to SGI to be able to smoothly animate a haptic world. Control commands from the SGI come at a very slow rate (0.01 ... 1.0Hz) and they are used only in special cases

when the user rotates or translates a visual and therefore also a haptic environment.

This architecture has several beneficial properties

- computationally heavy rendering processes can be separated into different computers that have adequate capabilities to handle the chosen processes
- haptic rendering of complex virtual prototypes is possible
- no need for a separate collision detection algorithm
- stereographic view is available
- short communication delays
- stable and accurate haptic rendering
- ability to haptically simulate static and kinetic frictions
- haptic smoothing is enabled.

The main weakness in this architecture is the need for two computers. The advantages are however so great that this approach was selected for further development. This architecture contains three processes as shown in Figure 25. There is one extended Open Inventor format file that is loaded by haptic and visual processes. Additional Open Inventor classes were created to be able to load

- haptic buttons
- surface materials
- haptic smoothing coefficient from a file.

The distributed nature of the architecture and inability to expand Ghost's Inventor format file loading capabilities required additional classes that enabled us to

- communicate with the visual server
- parse the haptic scene from an extended Open Inventor file
- easily rotate and translate haptic and visual environments synchronously.

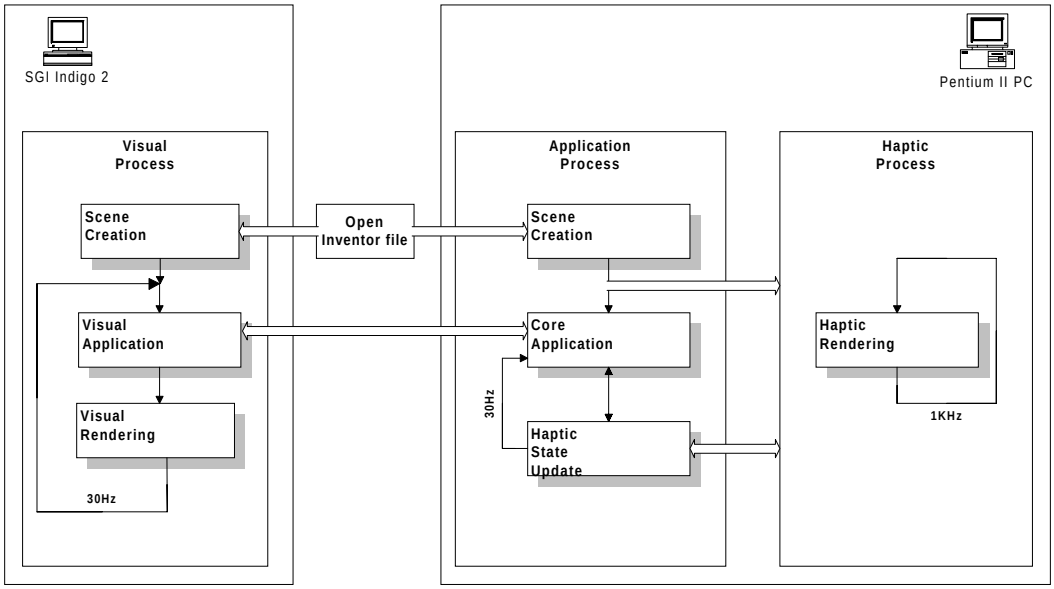

Figure 25. Processes of Visual server architecture.

Haptic rendering is started by loading an extended Open Inventor format file by application and visual processes. Next, the communication protocol is initialised between visual and application processes. Then, the visual process is blocked until the first position data is received from the application process.

The application process initialises the haptic process and reparses the haptic scene that has been loaded by Ghost. This reparsing is required for reading smoothing, materials and buttons from a file. When the haptic scene is improved with this additional data, the haptic process is started. The application process then sends position and orientation data about the user's fingertip and other objects to the visual process.

The haptic process has the highest process priority in the PC which means that the application process gets processing time when the haptic process grants it.

3.2.5 Comparison of different architecture alternatives

This chapter summarises previous architecture chapters by providing tables that present the used components and characteristics of different architecture

alternatives. Table 2 shows components that have been used with different architecture alternatives.

Table 2. Used software components.

SW component	*Fully distributed architecture*	*Haptic server architecture*	*Single workstation architecture*	*Visual server architecture*
Open Inventor	yes	yes	no/yes *)	yes
I-Collide	yes	yes	no	no
Armlib	yes	yes	no	no
Ghost	no	no	yes	yes

*) was not used at the tests, but is possible to use.

Table 3 shows the characteristics of different architecture alternatives. The characteristics are mainly defined by the software used and the hardware components.

The main weakness of the Fully distributed and Haptic server architectures was the inability to render complex-shaped virtual prototypes. The Fully distributed architecture also required too many computers to be convinient for use.

The weaknesses in the Single workstation architecture were the inability to render complex-shaped virtual prototypes and to use stereographic glasses. The first constraint was set by the processing power of the computer used. This architecture however looks very attractive for use with simple virtual prototypes.

Based on the properties shown above, the Visual server architecture was selected. The most important factor in the selection was that the Visual server architecture is the only architecture that provides a stable and accurate rendering of complex-shaped virtual prototypes with the computers that were available in the VRP project during the development of the haptic rendering system.

Table 3. Characteristics of architecture alternatives.

Characteristic	*Fully distributed architecture*	*Haptic server architecture*	*Single workstation architecture*	*Visual server architecture*
Number of used computers	3	2	1	2
Separate collision detection required	yes	yes	no	no
Separate communication method required	yes	no	no	yes
Separate plane selection algorithm required	yes	yes	no	no
Stereographic view available	yes	yes	no	yes
Support for stable haptic rendering of complex-shaped objects	no	no	no	yes
Haptic smoothing	no	no	yes	yes
Haptic buttons	yes	yes	yes	yes
Surface frictions	yes	yes	yes	yes
Virtual fingertip	yes	yes	no	no

4. EVALUATION OF THE IMPLEMENTED HAPTIC RENDERING SYSTEM

The main objective of haptic rendering in the VRP project was the ability to render a complex shaped electronic product. For this reason the Visual server architecture was selected for the implementation of the haptic rendering system.

The haptic rendering system implemented was evaluated with a pen-shaped cellular phone [19]. The Penphone was 14 cm long and its maximum radius was 1cm. It consists of five buttons and a display. The Penphone was constructed from 11 different pieces. Small details 12 and 13 are included in the upper cover and are discussed later. Different parts and the triangle count of each part are described in Table 4.

Figure 26. Virtual prototype of pen-shaped cellular phone.

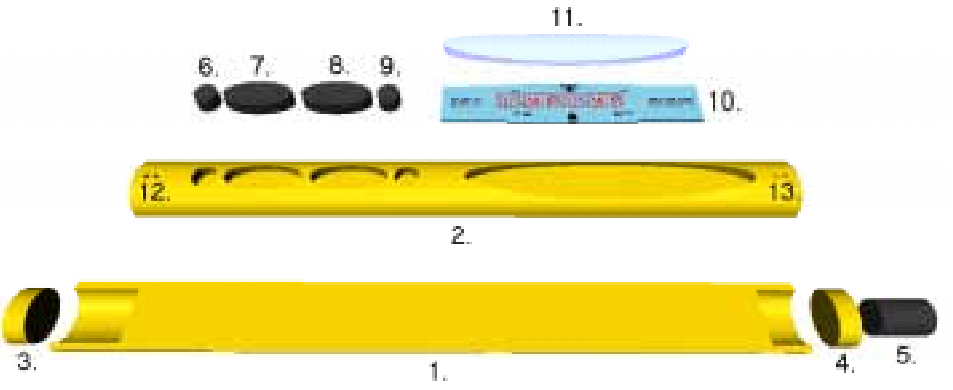

Figure 27. Penphone taken to pieces.

As Figures 26 and 27 show, the Penphone has been constructed from closed parts that are easily changeable. This separation into independent parts should be considered at the CAD design phase to ease the modification from a CAD model to a functional virtual prototype that can be haptically rendered.

Table 4. Penphone parts and the number of triangles in them.

Part	*Part Number*	*Number of Triangles in part*	*3D primitive*	*Material*
cover, lower part	1	364	face set	hard plastic
cover, upper part	2	1492	face set	hard plastic
left knob	3	64	cylinder	plastic
right knob	4	64	cylinder	plastic
button, head	5	64	cylinder	plastic
left small button	6	64	cylinder	plastic
left large button	7	64	cylinder	plastic
right large button	8	64	cylinder	plastic
right small button	9	64	cylinder	plastic
display board	10	60	6 x cube	plastic
display cover	11	132	face set	glass
microphone	12			
receiver	13			
Total		2496		

4.1 Haptic rendering model creation

To acquire full benefit from virtual prototyping with haptics, the creation and modification of haptic rendering models should be easy, convenient and quick. The creation of a virtual prototype can be started with a conventional CAD design program. To ease the transformation from a CAD model to a dynamic virtual prototype, the CAD model should be created from parts as in the Penphone (Figure 27, Penphone taken to pieces). This facilitates producing different virtual prototypes when different compatible parts can be tested.

In Figure 28 [19], the lower cover, knobs and display parts could be the same in all prototypes. Only the upper cover needs to be changed when creating different virtual prototypes so that different sized and shaped buttons can be fitted. In the early stages, buttons for different prototypes can be easily created from cylinders.

Figure 28. Variations of a pen-shaped cellular phone designed by JP Metsävainio Design Oy.

After creating the CAD model it must be converted to Open Inventor format. When the prototype is in Open Inventor format, the static prototype created can be haptically rendered with default material properties and without dynamics.

Once the virtual prototype's parts have been created, they should be stored in a component library. Then, when creating new virtual prototypes, the prepared components or, for example, haptic materials could be selected from the component library to quickly create new virtual prototypes.

Further development onto a prototype can be carried out with VpEditor [20] (Virtual Prototype Editor) which is a tool created at VTT Electronics. VpEditor is used for modifying Open Inventor format models and it provides two views of the virtual prototype (Figure 29).

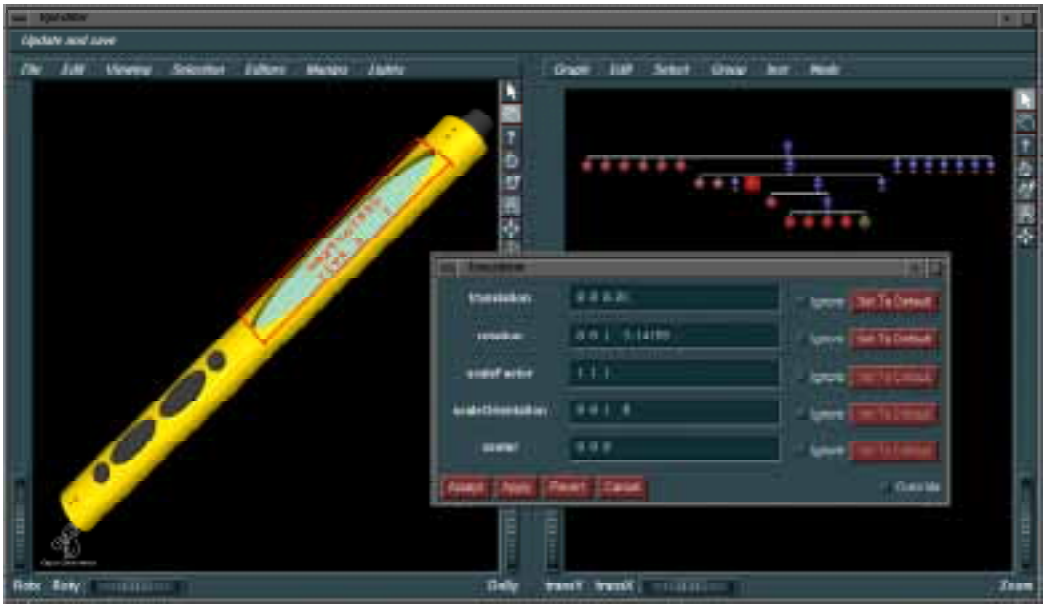

Figure 29. VpEditor view of penphone.

The left window shows the 3D visual representation of the virtual prototype. The prototype can be zoomed, rotated and moved with thumbwheels or a mouse. It provides also colour and material editors with which visual colours and materials of different parts can be easily changed and a ransformation editor which is used for scaling and repositioning parts.

The right window shows the internal representation of the 3D model. It is a hierarchical tree of extended Open Inventor nodes. The values of nodes can be edited to change for example the position or size of the parts. In Figure 29, the transformation node of the display cover has been selected for editing. This causes the display cover from both views to be framed with a red box and an edit box to be opened. Then new values can be inserted and applied to both scenes. New nodes can be created by selecting them from a menu. Nodes that can be created include all basic Open Inventor nodes and special nodes for

virtual prototyping which have been created in the VRP project. These nodes for example bring displays and haptic effects such as materials, friction, smoothing and buttons to virtual prototypes.

The VpEditor can be used for tuning the virtual prototype's visual and haptic properties. It is a very good tool for fitting parts of the virtual prototype together. Small details and therefore also small holes between virtual prototype's parts complicate haptic rendering. With the VpEditor, a virtual prototype can be easily rotated and zoomed so that even the tiniest holes can be noticed. Then parts can be moved and scaled to fill these holes. It is often better to scale adjacent parts so that they slightly overlap. This improves the stability of the haptic rendering system.

The whole process for creating and modifying a haptically renderable virtual prototype has been described in Figure 30. When the original CAD model has been constructed with the requirements of virtual prototyping in mind, the process from a CAD model to a functional virtual prototype can be very fast.

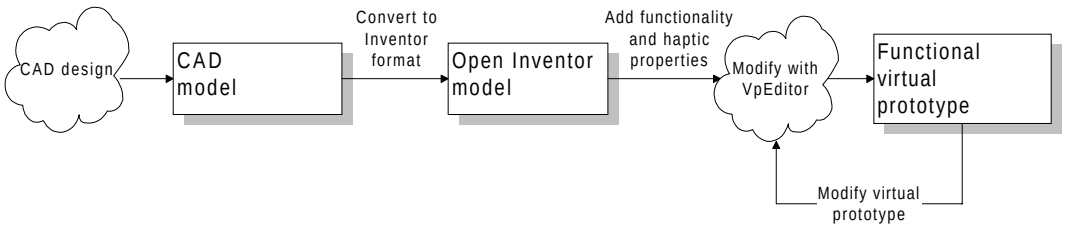

Figure 30. Virtual prototype creation process.

4.2 Use of the haptic rendering system

Figure 31 shows an example of how the user manipulates a virtual prototype with the Phantom haptic device. The stereoscopic view makes user see the simulated virtual prototype as floating in front of the screen.

Figure 31. Virtual prototype's usability test with a haptic device.

The visual user interface is shown in Figure 32. The visual user's fingertip has been presented as a small green ball. The user can feel force and tactile feedback when the green ball is in contact with the simulated virtual prototype. The user can feel the shape of the simulated virtual prototype and also materials of different parts. Buttons can be pushed and the display of the simulated prototype is updated depending on the buttons pushed.

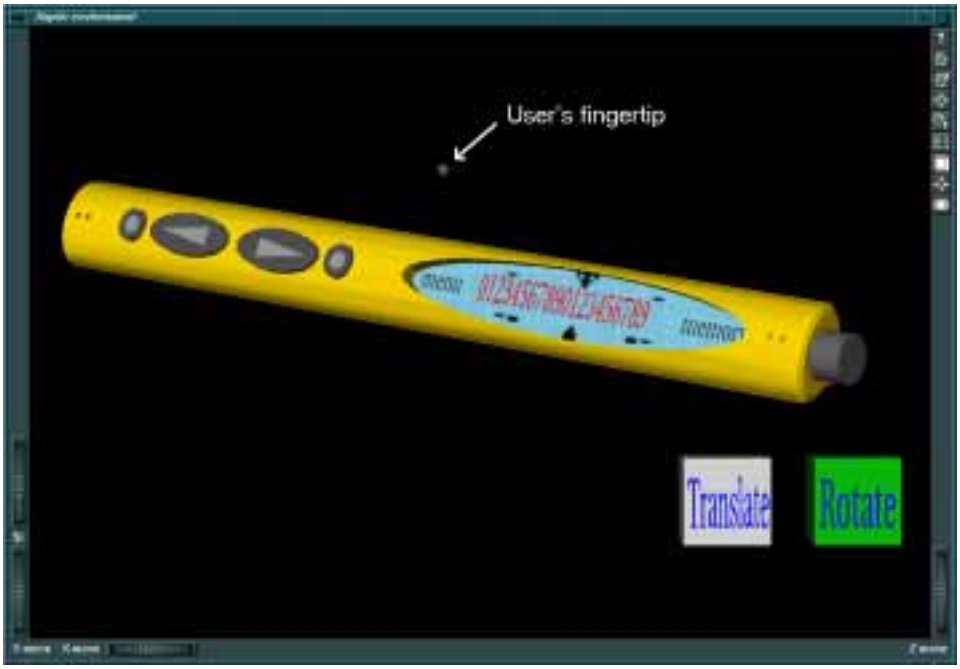

Figure 32. Visual user interface of haptic rendering (Text and arrow pointing to the user's fingerip has been added to original image).

The virtual prototype can be rotated and moved by first selecting the desired function with haptic Rotate and Translate buttons, and then activating/deactivating the effect with the spacebar. The selected function button is shown to be green on the screen (Rotate has been selected in Figure 32). Actual rotating and moving is handled with the haptic device. When the user moves his fingertip after activating the Translate effect, the simulated virtual prototype moves correspondingly. When Rotate has been selected, the user can rotate the prototype around its midpoint by moving his/her finger on a perimeter of a virtual ball whose center is at the midpoint of the simulated virtual prototype and radius is the distance between the midpoint of the virtual prototype and user's fingertip position.

In some cases, the user may get stuck inside the virtual prototype. In these cases, haptic rendering can be stopped, and the user is able to pull the fingertip out, by pressing the H key on the keyboard. Stopping the haptic rendering causes the user to be unable to feel the simulated virtual prototype but can see the

movement of user's fingertip. Then, when the user's fingertip is outside the virtual prototype, rendering can be continued by pressing the H key again.

Moving and rotating of a virtual prototype allows comfortable manipulation. The prototype can be adjusted to positions that allow the efficient usage of workspace of the haptic device used and also allows product usability tests when the simulated virtual prototype is in different positions.

At the moment, haptic rendering and editing of virtual prototypes are separated so that the rendering has to be stopped if the properties of a virtual prototype are changed. The aim is however to increase the usability of the system by allowing the user to dynamically change haptical parameters. The user could then select the part to be manipulated and then for example choose predefined materials such as wood and metal or define friction parameters to be applied to selected part.

4.3 Quality of the haptic rendering

Evaluation of haptic rendering quality means evaluation of what can be rendered and how realistic the rendered properties are. The quality of haptic properties cannot be measured accurately. When evaluating haptics, the sense of touch is the main indicator. Different people evaluate things differently but also experience in using a haptic device effects how the quality of the haptic rendering has been experienced. The evaluation of haptic properties is based on the writer's experiences and comments from several users.

Those haptic properties that can be simulated with the selected software and hardware architecture (Visual server architecture) are

- shapes
- stiffness of material
- static friction
- kinetic friction
- damping of material
- buttons
- sliders

- dials
- smoothing.

So what do these properties feel like? The first attribute to be evaluated is the stiffness of the surface. It is important that when the user sees a prototype that is to be made of hard plastic for example, the surface of the prototype also feels hard. When pushing hard on hard surface, the user should not perceive significant elasticity and should not be able to go through a virtual wall. When the Phantom haptic device is used correctly, surfaces feel hard, but if we only want to test the limits of the haptic device, it is possible to feel softness in a hard surface. This testing of the limits of the haptic device is usually connected only to the familiarising phase. After a little while the user tends to adapt to the capabilities of the haptic device.

The shapes of virtual prototypes can be accurately defined. Prototypes can be constructed from shape primitives (spheres, cubes, cylinders and cones) or more complex prototypes constructed from separate planes.

Smooth surfaces from separate planes can be implemented with a smoothing parameter. This parameter defines how the edge between separate planes is smoothed. The effect of this parameter feels realistic. Really smooth surfaces can be constructed with one constraint: smoothing tends to round also sharp corners that should not be smoothed. In the VRP environment, smoothing can be applied to separate parts. So with the Penphone prototype, it can be applied for example to the upper cover. In this case, sharp edges at the button holes also tend to be rounded. This causes drifting of the user's fingertip under the button where it can be trapped. The value of the smoothing parameter can be steplessly applied so that a balance can be achieved.

The surface frictions in the VRP project are constrained by the properties of the Ghost haptic library. It provides only two parameters for defining friction: static and kinetic friction coefficients. With so few parameters, accurate creation of different surface materials is problematic. However at the case of Penphone, very realistic sensations can be received when user moves his/her fingertip from the display of the penphone (glass material) to the cover of the penphone (plastic). The difference between materials is very clear and materials feel realistic.

4.4 Performance evaluation

The selected architecture was tested to find out how complex prototypes could be properly rendered. Ghost automatically stops execution of the haptic loop if it takes too much time, which means that the stability of the haptic rendering cannot be maintained. Knowing this property, the maximum complexity of the haptically rendered prototype can be determined.

Communication frequency from the haptic process to visual process was measured to be able to figure out what is the visual update rate at different prototype complexity rates.

Table 5. Performance evaluation.

Number of Triangles	*Haptic rendering rate [Hz]*	*Communication frequency between visual and haptic processes [Hz]*
96	967-1130	21.25 - 32.25
2496	960-1125	21.25 - 32.25
6194	955-1121	21.25 - 32.25
12388	870-1100	21.25 - 32.25
18582	0-1013	21.25 - 32.25

The communication frequency between the visual and haptic processes seemed to be the same at all complexity rates (Table 5). As can be seen, the complexity of the rendered prototype effects on the haptic rendering rate. The maximum number of triangles in the prototype does not, however, tell the whole truth about the performance of the rendering environment. Although it seems that the architecture used can render the Penphone (2496 triangles) easily, there are some problems. The smaller the details in a virtual prototype are, the easier the problems occur. When rendering small details of the most complex prototype constucted from 18582 triangles, the stability of the haptic rendering could not be maintained. Also the differences between maximum and mimimum rendering rates at certain complexity level are caused by the same fact. The rendering rate decreases when small details are rendered.

In the case of a Penphone, the tiny holes of the microphone and receiver caused problems (Figure 27). When moving one's finger in these small holes it was possible to end up inside the prototype. This was caused by the fact that these small holes are also constructed of virtual planes and when the user moves his/her fingertip too fast, the collision detection algorithm fails to report collision with the plane on time and the user is able to pass the surface before the haptic rendering sets the surface active. Then, because the virtual prototype of penphone is hollow, the user is trapped inside the virtual prototype. To get out, the user deactivates the haptic rendering from the keyboard and after getting out activates the haptic rendering again.

The physical size of a prototype effects on maximum renderable complexity of a virtual prototype. Large prototypes are easier to render than equal-shaped smaller prototypes. This is caused by the fact that when the user's fingertip moves on a virtual surface constructed from small planes, virtual planes must be updated more frequently and so the collision detection algorithm has also to be executed faster.

5. DISCUSSION

At the time of writing this thesis, the experimental haptic rendering system has not yet been used in large-scale industrial product development projects. The first projects are about to begin so that industrial experiences about the benefits of applying haptic rendering to product development are only preliminary. The rendering system implemented was however tested by several test users with varying technical backgrounds. Based on the feedback from these users, preliminary results about the realism achieved in the user-to-product interaction are reported.

5.1 Haptic rendering experiences

The haptic rendering system implemented has been tested by several users with varying technical background. Experiences vary on how realistic the haptic rendering feel like. Some users get very good sensations about the haptic rendering and are very enthusiastic and fascinated about the possibility to touch and feel 3D models. These users think that the rendered virtual prototype is very realistic. However, for some novice users perceiving 3D objects with this visual rendering technique may cause some trouble, and this leads to incorrect use of haptic device and therefore the haptic sensations cannot seem realistic. Between these extremes is the majority of users but it seems clear that they are very much closer to the enthusiastic and fascinated group.

As with any device, usage of the haptic device may require a little practise for a user to be able to fully understand and get the full benefit from haptic rendering. Some novice users may push virtual prototypes too hard when the user's hand is not relaxed or when the prototype has not been understood as a three dimensional object. The opposite of this is that the user believes that he/she is touching something even when the user's finger is far from all surfaces. Some users say that the Phantom's own mass and inertia decrease the realism of haptic sensations. After a little practising, manipulation of the virtual prototype the with haptic device becomes more natural and the dimensions and physical properties of the device and simulated prototype become clearer to the user. Common enthusiasm and interest towards haptics also tends to give the user better sensations from haptic rendering.

Experiences showed that realistic virtual prototypes could be created. The creation of buttons was easy and the properties of buttons were easily changeable. Creating realistic material sensations was more complicated. At least realistic sensations of glass and plastic could be simulated. Shapes of virtual prototypes could be accurately defined, but the processing power of the computers used limited the minimum size of the details that could be accurately rendered.

5.2 Benefits

As described in the beginning of this study, making physical prototypes is expensive and time consuming. Physical prototypes are used for evaluation of the physical properties of the prototype (shape, materials, user-interface). Virtual prototyping provides a quick and cheap way of creating fully functional prototypes and modifying them. Haptic rendering increases the realism achieved by providing touch interaction with virtual prototypes.

When visual and haptic environments are synchronised so that no perceivable delays occur, haptic rendering with a stereographic view provides a very powerful means for product usability tests. Product testing gets much more realistic when the user is able to feel the product and push its buttons and feel the spring forces and at the same time see the behaviour. In this way it is easier to understand the products behaviour more thoroughly.

Similarily future products that can not be manufactured with the current technology, can be designed and tested with virtual prototypes.

5.3 Problems

The stability of the implemented haptic rendering system is good when simulating prototypes such as the Penphone. Only the smallest details cause a little instability. These problems could be solved with the virtual fingertip.

The Phantom haptic device constrains the realism of sensations that can be received by using a single thimble as a user interface. This does not allow the most natural usage of small handheld electronic devices since the user is not able to hold the device in his/her hand but only to manipulate it with one finger. Despite this fact, the Phantom provides accurate haptic sensations for one finger. When performing product usability tests, one finger is in most cases sufficient for creating a realistic illusion of the user-to-product interaction.

The performance of the computers used constrains the virtual prototype's shape-complexity. This leads to finding distributed architectures that could provide efficient haptic rendering without too long communication time delays between the graphical and haptic rendering processes. The selected architecture for the haptic rendering system seems to provide a good approach. There are no loops that should be executed fast so that the communication frequency between the separate computers is quite low.

5.4 Further development

The development of the presented haptic rendering system continues. One task is to find realistic haptic rendering attributes to simulate materials for virtual prototypes. At the time of, a virtual fingertip was under development. The virtual fingertip will be implemented soon and the usability of the rendering system will be improved by elaborating its user-interface. New features will be introduced to allow interactive modification of haptic properties during haptic rendering.

One way of improving the realism of haptic rendering could be a combination of real and virtual worlds (Figure 33). In this case, the user can feel the virtual prototype with the Phantom haptic device, as described previously, but also use the other hand with a data glove for defining the orientation and position of a virtual prototype by moving and rotating a smooth physical mock-up. This would allow the user to hold a virtual prototype in his/her hand rather that see it floating in the air. Buttons, materials and surface shapes could be virtually

created on the surface of the physical mock-up with the haptic rendering technique. This technique would also require the use of a HMD.

Figure 33. Real and virtual worlds combined. 1) Data glove, to simulate virtual hand and to accurately measure position and orientation of the user's hand and fingers. 2) Physical mock-up, where position and orientation can be measured with trackers. 3) Virtual prototype, which is overlapped with physical mock-up. 4) The Phantom haptic device.

Searching for new software and hardware components to improve implemented haptic rendering system continues. At the moment, no new useful software components have been found. Hardware component updating may be concentrated on obtaining more powerful workstations and improving visual rendering. It is clear that as the performance increase of computers continues, also the rendering capabilities of the implemented haptic rendering system will improve.

At the moment there are also no haptic devices on the market that would allow force and tactile feedback for all fingers. Presumably these devices will be available in the future when the number of delivered haptic devices increases, prices of haptic devices drop and the technology develops.

6. Summary

In this study a haptic rendering system for the VRP virtual prototyping environment was implemented. Four different architectures for a haptic rendering system with the presented software and hardware components were described. Based on the properties of architectures, the Visual server architecture was selected. It separates the visual rendering process onto another computer from the other processes, which seems to be the best approach when aiming for short communication delays between processes. This allows stable and accurate haptic rendering of complex-shaped virtual prototypes, which was the primary goal. The software components in this architecture also enable convenient creation of various haptic properties.

The implemented haptic rendering system was tested with a virtual prototype of a pen-shaped cellular phone. Results showed that the rendering system was able to simulate the selected prototype efficiently with the exception of minor instability when rendering the smallest details. The process of creating a haptic rendering model from a 3D model was based on converting a CAD model to Open Inventor format and adding the functionality and the haptic properties to this model with the VpEditor tool. Experiences showed that creating a haptically renderable prototype from a static Open Inventor format model and modifying it with the VpEditor is easy and quick.

The haptic rendering system separated haptic rendering and modification of a virtual prototype. The haptic sensations created were quite realistic, taking into account the properties of the haptic device and the graphics rendering system used. Experiences on the selected haptic device, Phantom, showed that its use may require a little practising for a user to be able to get full benefit from haptic rendering. Also the realism achieved from the haptic rendering was constrained by the simplicity of the one-finger user interface of the haptic device. However, in most cases users were able to quite easily adapt to the properties of the Phantom haptic device.

The further development of the system is focused on increasing the usability and improving the realism of sensations. One way of improving the usability is to partially combine haptic rendering and haptic model modification environments to speed up the process of testing different haptic parameters with virtual

prototypes. Implementation of a virtual fingerip, which enables simulation of various finger types, would increase the usefulness of haptic rendering by providing more informative usability tests. To increase the realism of the implemented haptic rendering system, a combination of real and virtual worlds could be used. This would, for example, allow the user to pick up and hold a virtual device in one hand and press the buttons with the other hand utilizing the haptic device.

References

[1] Kerttula M., Salmela M. & Heikkinen M. (1997) Virtual Reality Prototyping - a Framework for the Development of Electronics and Telecommunication Products. In: Proceedings of the 8th IEEE International Workshop on Rapid Prototyping, June 24-26, 1997, The Carolina Inn-Chapel, North Carolina. IEEE Computer Society, Los Alamitos, California, 10 p.

[2] Allen F. (1996) Adding Texture to Force-Feedback Displays: A Review of Recent Research. University of Oulu, Department of Information Processing Science, HCI & Group Technology Laboratory, 108 p.

[3] Burdea G. (1996) Force and Touch Feedback for Virtual Reality. John Wiley & Sons, Inc., New York, USA, 339 p.

[4] Mark W., Randolph S., Finch M., Van Verth J. & Taylor II R. (1996) Adding Force Feedback to Graphics Systems: Issues and solutions. University of North Carolina at Chapel Hill, USA, 6 p.

[5] VRP project at VTT Electronics, http://www.ele.vtt.fi/projects /vrp/vrp.html (October 20, 1997) Copy available from author.

[6] Kaksonen R. (1997) Programmer's Reference for the VRP Communication System, VRP Project Report, VTT Electronics, Oulu, Finland, 41 p. (Copy available from author)

[7] Ruspini D., Kolarov K. & Khatib O. (1997) The Haptic Display of Complex Graphical Environments. In: Computer Graphics Proceedings of the SIGGRAPH 97. ACM SIGGRAPH, New York, USA, pp. 345-352.

[8] Mark W., Randolph S., Finch M. & Van Verth M. (1996) UNC-CH Force-Feedback Library, Revision C.2. University of North Carolina at Chapel Hill, 54 p.

[9] Anonymous (1997) Ghost Software Developer's Toolkit, Programmers Guide, Version 1.00. SensAble Technologies, Inc., Cambridge, MA, USA, 40 p.

[10] Massie T., Salisbury J. (1994) The Phantom Haptic Interface: A Device For Probing Virtual Objects. In: Proceedings of the ASME Winter Annual Meeting, Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems, Chigaco, November 1994. Paper available from SensAble Technologies, Inc., Cambridge, MA, USA, 7 p.

[11] Phantom haptic device, http://www.sensable.com (October 20, 1997) Copy available from author.

[12] Cohen J., Lin M., Manocha D. & Ponamgi M. (1994) I-COLLIDE: An Interactive and Exact Collision Detection System for Large-Scale Environments. University of North Carolina at Chapel Hill, USA, 8 p.

[13] Wernecke J. (1994) The Inventor Mentor. Addison-Wesley Publishing Company, Reading, Massachusetts, USA, 514 p.

[14] Wernecke J. (1994) The Inventor Toolmaker. Addison-Wesley Publishing Company, Reading, Massachusetts, USA, 301 p.

[15] Lin M. (1993) Efficient Collision Detection for Animation and Robotics. Department of Electrical Engineering and Computer Science, University of California at Berkeley, CA, USA, 139 p.

[16] Anonymous. I_COLLIDE USER'S MANUAL - Release 1.1. University of North Carolina at Chapel Hill, USA, 7 p.

[17] Sinha A. (1996) Network programming in Windows NT / Alok K. Sinha. Addison-Wesley Publishing Company, Reading, Massachusetts, USA, 620 p.

[18] Intergraph graphics workstations, http://www.intergraph.com/ (October 25, 1997) Copy available from author.

[19] Komulainen O. (Forthcoming) Matkapuhelimen langaton käyttöliittymä. Masters Thesis. University of Art and Design Helsinki, UIAH. (Taiteen Maisterin tutkinnon lopputyö. Taideteollinen Korkeakoulu, TAIK, Helsinki

[20] Kyllönen H. (1996) Developing Open Inventor-based Virtual Prototypes. VRP Project Report, VTT Electronics, Oulu, Finland, 16 p. (Copy available from author)