Design and development of a framework to bridge the gap between real and virtual

by

SK Alamgir Hossain

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements
For the Master degree in
Computer Science

Ottawa-Carleton Institute for Computer Science
School of Electrical Engineering and Computer Science
Faculty of Engineering
University of Ottawa

Abstract

Several researchers have successfully developed realistic models of real world objects/ phenomena and then have simulated them in the virtual world. In this thesis, we propose the opposite: instantiating virtual world events in the real world. The interactive 3D virtual environment provides a useful, realistic 3D world that resembles objects/phenomena of a real world, but it has limited capability to communicate with the physical environment. We argue that new and intuitive 3D user interfaces, such as 3D virtual environment interfaces, may provide an alternative form of media for communicating with the real environment. We propose a 3D virtual world-based add-on architecture that achieves a synchronized virtual-real communication. In this framework, we explored the possibilities of integrating haptic and real world object interactions with Linden Lab's multiuser online 3D virtual world, Second Life. We enhanced the open source Second Life viewer client in order to facilitate communications between the real and virtual world. Moreover, we analyzed the suitability of such an approach in terms of user perception, intuition and other common parameters. Our experiments suggest that the proposed approach not only demonstrates a more intuitive mode of communication system, but also is appealing and useful to the user. Some of the potential applications of the proposed approach include remote child-care, communication between distant lovers, stress recovery, and home automation.

Acknowledgements

This thesis is based on research conducted during my years at the Multimedia Communications Research Laboratory (MCRLab) and the Discover Lab, University of Ottawa. The work presented here is the result of highly collaborative efforts and I owe thanks to the people I have worked with.

First and foremost, I would like to express my sincere gratitude to my supervisor Prof. Dr. Abdulmotaleb El Saddik, for giving me the opportunity to pursue research on this rapidly emerging field and move forward along the path of innovation and novelty. His invaluable research ideas and experience kept me motivated all the way towards the progress and completion of this research work.

Besides my supervisor, I would like to acknowledge my colleague Abu Saleh Md Mahfujur Rahman for the collaboration and discussion, and implementing a portion of the prototype. I would also like to thank my other colleague Anwar, Anis, Shamim and Suruz Miah for their valuable source of information and inspiration.

My parents, SK Habibur Rahman and Mrs. Anwara Rahman, have been a continuous source of love, support and encouragement all throughout my graduate studies. Last but not the least, I am grateful to my dearest wife, Tamanna Tabassum, for her belief and encouragement to follow my dreams ...

Dedication

I dedicate this thesis to my parents ...

Contents

List of Tables

List of Figures

Glossary of Terms

2D	Two Dimensional
3D	Three Dimensional
API	Application Programming Interface
AW	Active World
BVH	Biovision Hierarchy
CMake	Cross Platform Make
FCFS	First Come First Serve
GUI	Graphical User Interface
HUD	Heads Up Display
IDE	Integrated Development Environment
L. Lab	Linden Lab
MFC	Microsoft Foundation Class
PS-H	Playstation Home
SAPI	Speech API
SDK	Software Development Kit
SL	Second Life
UI	User Interface
VE	Virtual Environment
VR	Virtual Reality
WoW	World of Warcraft
XML	eXtensible Markup Language

Chapter 1

Introduction

1.1 Background and Motivation

Plato introduced the concept of virtual reality in 380 BC, through the Allegory of the Cave in his famous work, 'The Republic' [21]. Technology has realized Plato's Cave through a three-dimensional (3D) virtual environment. A 3D virtual environment, commonly known as a Networked Virtual Environment (NVE), is a computer simulated environment that reproduces a real world, wherein humans control their activities inside the world through their digital representations, termed 'avatars'. The avatars are controlled though the uses of a keyboard and/or mouse and in most cases have limited capability to communicate with real-world objects. Besides avatars, virtual environments contain objects found in the real world, such as buildings, trees, and furniture. Although both avatars and objects have commonalities, they have different behaviours and purposes. Avatars are generally non-persistent entities, i.e. the avatar leaves the virtual world when the user's computer is disconnected from it. On the other hand, objects are generally static, persistent, and non-movable with some exceptions like cars, bullets, and arrows etc.

One of the most popular and rapidly spreading examples of a 3D virtual environment is Linden Lab's *Second Life*[1][7, 27, 38]. In Second Life, similar to *ActiveWorlds*[2] and *Sims Online*[3], once connected the users can view their avatars in the 3D virtual environment; they can then participate, in real-time, in task-based games, play animation, and communicate with other avatars through instant messaging and voice. Moreover,

[1]Second Life, http://secondlife.com

[2]Activeworlds Inc, http://www.activeworlds.com

[3]Sims Online, http://www.ea.com/official/thesims/thesimsonline

the flexible authoring capabilities of 3D objects [20, 19], their built-in security mechanism, scalability, and portability features make Second Life the most popular 3D virtual environment. Currently it has 25 million active subscribers, as of August 2011 [30].

In the future, virtual environments will be more intuitive and interactive, like the real world, because of advances in computer 3D video rendering technology. The huge impact of the 3D virtual environment has been perfectly described by *Gartner*, an information technology research and advisory firm. Gartner has stated that the growing demand for virtual reality will increase significantly by the end of 2011 and that almost 80% of Internet users will have an account in the virtual environments [17].

The 3D virtual environment proves to be a useful and perfect simulation of the real world. However, it has limited capability to communicate with real world objects. A mode of communication that can enhance the user experience is to incorporate the sense of touch, which has a great deal of significance in inter-human communication. These social/emotional tactile experiences, in the form of handshakes, encouraging pats, hugs, tickles and other physical contacts, are fundamental to mental and psychological development; hence their applications in interpersonal communication systems have attracted the attention of many researchers around the world [18]. Driven by this motivation, we explored the possibilities of integrating haptic and real world object interactions with Linden Lab's multiuser online 3D virtual world, Second Life. We enhanced the open source Second Life viewer client in order to facilitate the communication of emotional feedback, such as human touch, encouraging pats and comforting hugs to the participating users through real-world haptic stimulation.

Considerable efforts in the past few years have aimed at bridging the gap between the real and virtual environments [6][16][39][44][40]. Researchers around the world are aiming to leverage the sense of touch in communication media between multiuser 3D virtual environments and the real world. Since the sense of touch has major significance in inter-human communication, and to convey emotional feedback, the haptic is given high regard both in live communication [8][49] and in immersive virtual environments [12]. The haptic-based nonverbal modality can enhance social interactivity and emotional immersive experiences in a virtual world that presents a realistic 3D environment, where people can enrol in an online virtual community [45].

1.2 Research Problem and Direction

In the physical world, a range of wired and wireless sensory devices such as temperature, humidity, and pressure sensors are available. By using these sensory data it is possible to automate and control various physical devices to create a soothing environment. The prices of these sensors and automation devices (such as X10[1] and *Wi-Fi*) are decreasing. At the same time, their utility in providing control and various entertainment facilities in smart spaces are making them increasingly popular. A graphical user interface (GUI), which is a medium with which to interact with programs, plays an important role in controlling physical space elements like smart homes. However, controlling a physical space by using the traditional icon-based GUI might not be sufficient for effective management of a range of smart physical devices. In order to efficiently control different intelligent physical devices, researchers have focused their attentions on the intuitive GUI design and explored its usability issues [6] [36]. Spatial cognition studies have shown that navigating through 3D virtual interfaces can assist the user and help better perceive the environment [50, 22, 2]. 3D virtual interfaces also permits the user to access information faster [11]. In our research, we leverage the advantages of 3D virtual interfaces and propose a framework to control and visualize the real sensory data from the smart physical environment.

Integrated specialized communication services have become an appealing application for networked virtual environments (e.g. Second Life). With specialized communication services, people can send customizable haptic or device-control data over the Internet, which can play a synchronized animation in the virtual environment that mimics the real world object's behaviour. This is a very challenging task, since the architecture needs to deal with the different requirements and characteristics of the avatars. Moreover, when the number of users becomes huge, the interactions will be increased dramatically, since millions of users may be interacting in a networked virtual environment. For this reason we chose add-on architecture that can work on top of existing virtual environments (like Second Life, WoW, etc) and can provide different services with which to control physical devices. Currently most of the virtual environments have features allowing for incorporation of custom scripts to change the normal behaviour of an object. We use this feature to create communication between the virtual world and the real world. We introduce an annotation mechanism by which a user can add access control lists to control the behaviour of an object. Moreover, with this enhanced version of customized

[1] *X*10, http://www.x10.com/homepage.htm

annotation mechanism, the user can control real world home appliances like lights, fans, music systems, etc. Finally, the architecture provides a communication metaphor that can generate personalized remote haptic signals that can be specially useful for remote lovers or for remote child caring.

Gaming is a possible way of intimate communication. Considering this assertion, there is no specific system that attempts to consider virtual environment as a kind of useful user interface that can help the human visualization and decreases the mind overhead. We particularly focus on this direction and leveraged existing methodologies in order to deliver a general architecture to bridge the communication between real and virtual environment.

1.3 Objectives and Contributions

The increasing demand for collaboration between the physical world and the virtual world inspires the researcher to do more intuitive and interactive work. The objective of this research is to enhance the typical collaboration between virtual environments and real world, in order to impart more realistic feelings to the user. Currently, with the help of the virtual world, users can navigate their avatar in the world, manage their virtual objects, and communicate with the other users through chat, voice etc. We designed and developed a framework with the requirement that it should be expandable. More specifically, it should be possible to add new modes of communication that can help the user to control not only virtual objects, but also real world objects. With the help of this framework, we developed a prototype system that can be used either for an interpersonal communication system or to control physical objects in the real world.

The goal of this thesis is to design and implement a framework to facilitate avatar-based real world communication. Overall, the objectives of the thesis can be summarized as follows:

- Finding a way to facilitate communication between the virtual and real world.
- Creation of an intuitive, user-friendly and flexible 3D annotation mechanism to actuate the communication between virtual objects and the associated real world objects.
- Incorporate haptic interaction between remote users through the virtual world.

The research in this thesis has been performed by leveraging several existing techniques on virtual environments and 3D user interface design mechanisms. However, most of the formulation, implementation, and experimentation found in this thesis consist of original work. The contributions of this thesis provide a flexible annotation mechanism for minimizing the gap between real and virtual environments, and as a whole can be summarized by the following:

- Design of a framework to facilitate interpersonal communication that proposes haptic interaction modality between real users and their respective virtual avatars through a flexible annotation mechanism in 3D virtual environments such as Second Life.
- By leveraging the proposed framework we designed a 3D object authoring based smart home automation scheme in Second Life.

1.4 Scholarly Output

In addition to meet its objectives as described above, this research undertaking has also lead to a variety of scholarly publications, as listed below.

a. **Papers in Refereed Journals**

[1] SK Alamgir Hossain, ASM Mahfujur Rahman, and Abdulmotaleb El Saddik, "Measurements of Multi-modal Approach to Haptic Interaction in Second Life Inter-personal Communication System," in *IEEE Transactions on Instrumentation and Measurement*, vol.60, no.11, pp.3547-3558, Nov. 2011.

b. **Papers in Referred Conference Proceedings**

[2] SK Alamgir Hossain, ASM Mahfujur Rahman, and Abdulmotaleb El Saddik, "Bringing Virtual Events into Real Life in Second Life Home Automation System," in *The IEEE Virtual Environments, Human-Computer Interfaces, and Measurement Systems*, 19-21 September 2011, Ottawa, Canada.

[3] ASM Mahfujur Rahman, SK Alamgir Hossain, and Abdulmotaleb El Saddik, "Bridging the Gap between Virtual and Real World by Bringing an Interpersonal Haptic Communication System in Second Life," in *The IEEE International*

Symposium on Multimedia (ISM), 13-15 December 2010, Taichungt, Taiwan, pp. 228-235.

[4] SK Alamgir Hossain, ASM Mahfujur Rahman, and Abdulmotaleb El Saddik, "Haptic based emotional communication system in second life," in *2010 IEEE International Symposium on Haptic Audio-Visual Environments and Games (HAVE)*, 16-17 October 2010, Phoenix, Arizona, USA, pp. 1-1.

[5] SK Alamgir Hossain, ASM Mahfujur Rahman, and Abdulmotaleb El Saddik, "Interpersonal haptic communication in second life," in *IEEE International Symposium on Haptic Audio-Visual Environments and Games (HAVE)*, 16-17 October 2010, Phoenix, Arizona, USA, pp. 1-4 (Demonstration).

c. **Submitted Journals**

[6] ASM Mahfujur Rahman, SK Alamgir Hossain, and Abdulmotaleb El Saddik, "Real Life Events Acquisition from Second Life Virtual Events to Bridge the Gap Between Virtual and Real in Smart Home Automation System," in *Multimedia Tools and Applications*.

1.5 Thesis Organization

The remainder of the thesis is organized as follows: Chapter 2 presents an overview of the different 3D virtual environments and relevant emotional communication systems. This includes a brief introduction of 3D avatars, linden scripting, Biovision Hierarchy (BVH) file structure, vibrotactile actuators, X10 protocol, and web service technologies. Chapter 3 proposes our idea and elaborates on the system design phase that includes the overall architecture of our proposed system, the Use Case model, proposed system features, and the working process of its different modules. Chapter 4 focuses on the implementation of the system. This includes our choice of technology, software architecture, sample user interfaces and the development process of different modules. Chapter 5 covers the evaluation and result of the experiments that we conducted in our laboratory environment using our prototype system; in this chapter, our primary focus is on user centric evaluation. Finally, the thesis concludes in Chapter 6 with suggestions for future research.

Chapter 2

Background and Related Works

2.1 Literature Background

In this chapter we present a review of the literature on 3D virtual environments. Here our motivation is twofold: the first is to describe the current popular virtual environments; the second is a comparative study of those environments. This thesis attempts to bridge the gap between virtual and real environments, so at the outset, the notion of the 3D virtual environment and its different types are introduced. This is followed by an overview of 3D avatars, linden script, BVH file structure, haptic jacket interface, and X10 protocols.

2.1.1 3D Virtual Environment

3D Virtual Environment (VE), according to *Bishop*, "is a genre of online community that often takes the form of a computer based simulated environment, through which users can interact with one another and create objects" [4]. As this is a very broad definition, we would like to narrow it down. A 3D virtual environment can also be defined as: an online Massively Multiplayer Online Game, commonly known as MMOG, which renders in 3D and produces a computer generated world that allows for multiple types of activities, like customizable avatars and 3D objects, text/speech based communication, and many more. The avatars can travel between cities, villages, and even virtual worlds to carry out business or leisure activities. Although the 3D virtual world is primarily for gaming, it also employs live video conferencing, text or voice based chat communication; even 'emoticons' or 'smilies' are available, with which to express feelings.

Second Life

Second Life (*SL*)[1] is the most popular online 3D virtual world, and was developed by Linden Lab[2] on 2003. Currently it has 25 million active subscribers, as of August 2011 [30]. It provides a free client program called *viewer*, allowing for interaction with one another through 3D avatars called *resident*. A subscribed user can enter into a *region* of *256*×*256* meters in size by clicking on the virtual map, which is commonly known as 'teleport to a map'. Like other virtual environments, each Second Life resident has a limited visibility area, called an Area of Interest (*AoI*). This is the maximum distance between two communicating avatars. The avatars can automatically receive all the messages and events within this area. Second Life avatars have the ability to walk, run, and even fly inside the worlds, from one region to another. Second Life land can be either public or private (see figure 2.1). Public lands are maintained by Linden Lab itself, but units of private land have to be maintained by the user who purchased the land from Linden Lab. Although private lands are maintained by the respective land users, both types run within the Linden Lab servers.

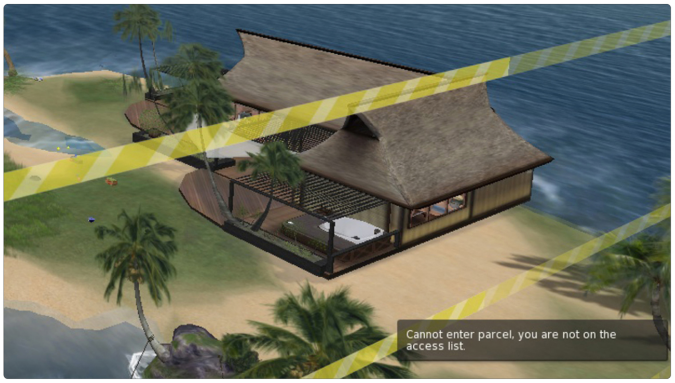


Figure 2.1: A private land in Second Life. The yellow line indicating the land boundary. Before entering a private land the user needs an approval from the land owner.

Linden Lab provides both a commercial and an open source version of the *SL viewer*. The open source viewer is called *snowglobe*[3], which they release under a GNU GPLv2 license. The viewer allows its users to create virtual objects through a 3D modeling tool. It also allows them to add more functionality to these virtual objects by incorporating

[1]Second Life, http://secondlife.com
[2]Linden Lab, http://lindenlab.com
[3]SnowGlobe, http://wiki.secondlife.com/wiki/Open_Source_Portal

rules based on its own scripting language, called Linden Scripting Language (*LSL*)[1]. Like other virtual environments, Second Life has its own economy and a currency system of Linden Dollars (*L$*), where *L$1,000 = $3.8 USD*. A comprehensive study of SL and the traffic it generates are found in references [1] [25] [26]. Currently SL is the most popular VE; two main reasons behind this popularity are: first and most importantly, it is free with all its facilities, whereas other virtual worlds have subscription fees ranging from USD $20 to $200 (depending on payment plans and features) [5]. Finally, SL gives the user intellectual property rights to what they create (see the comparison in table 2.1).

Figure 2.2: World of Warcraft: the landing screen of Cataclysm game [5].

World of Warcraft

World of Warcraft (*WoW*)[2] is a multi-player online role-playing game created by Blizzard Entertainment[3] in 2001. *WoW* is one of the popular online paid multi-player games. Currently it has 12 million active subscribers, as of October 2010 [15]. As in other virtual environments, each *WoW* subscriber has an avatar to navigate inside the virtual environment; additionally, the avatar must stay alive by fighting with various monsters. World of Warcraft is not free; users have to subscribe to play. Users can purchase prepaid game cards for a selected amount of playing time, or by using a credit or debit card to pay on a regular basis. Blizzard Entertainment divided the *WoW* into two separate packs: one is the core pack and the second is the expansion pack that can work as an add-on to the core pack. Blizzard Entertainment releases new expansion packs at different stages. Figure 2.2 depicts 'Cataclysm,' which is the *WoW's* fourth expansion pack.

[1]Linden Scripting Language Portal, http://wiki.secondlife.com/wiki/LSL_Portal
[2]World of Warcraft, http://us.battle.net/wow/en
[3]Blizzard Entertainment, http://us.blizzard.com/en-us/

Sony Playstation Home

Sony Playstation Home[1] is a *PlayStation*3 console-based virtual environment developed by Sony Computer Entertainment Inc. In order to enter into this virtual environment, users need a *PlayStation*3 console and an account (which is free) in the PlayStation Network (*PSN*). PSN users can create custom avatars, decorate their homes with default, bought, or won items. Moreover, they can navigate inside the virtual world, and the world is constantly updated by the users. In the virtual world, multiple game zones offer activities, streaming video and live events. Playstation Home is also becoming popular because the console has 3D graphics-rendering capabilities and voice-based interaction facilities. Currently, it has over 12 million subscribers (see table 2.1.1).

Figure 2.3: A popular place called "Central Plaza" in Sony Playstation Home.

Activeworlds

ActiveWorlds[2] is an older but nevertheless popular virtual world, developed and published by Active Worlds Inc. in the year 1997. Like other VEs, ActiveWorlds avatars can navigate, play online games, communicate with others, and make friends, among many other activities. Anonymous subscription is free, but people have to pay to access all the facilities. They provide two different membership plans: one is short-term subscription and the other is an unlimited time membership or citizenship plan. The ActiveWorlds citizen can access all facilities, including unlimited access, a protected name, and private property, access to objects and an avatar gallery, and many more.

[1]Sony Playstation Home, http://us.playstation.com/psn/playstation-home
[2]ActiveWorlds, http://www.activeworlds.com

The Sims Online

The *Sims Online* is an online massively multiplayer 3D virtual world developed by Electronic Arts in the year 2002 for the Microsoft Windows platform. Each subscribed user has an avatar called *Sims* in the virtual environment. After successful subscription, the user gets a virtual piece of land on which to build virtual objects such as a house, building, bar, club, or anything related to our daily life. The Sims can explore the vast user-created world, socialize, meet new friends, and develop their own reputation, power and social standing. Unlike other VEs, Sims have very human-like qualities, such as the need to sleep, eat, and hang out with people. Although the user has no fixed objectives or direct guidelines, the game has some unique features that correspond with the real world such as people who are developing their skills, or working as an employee to earn virtual currency (called 'simoleans'). Users can make more and more 'simoleans' and can purchase virtual goods and household items; or they simply can convert the virtual currency into real currency and receive these real dollars, provided that the users have to supply their bank account information. Electronic Arts started this project for testing purposes; this game is currently not running, as all its activities were shutdown on August 1, 2008.

Summary of 3D Virtual Environments

In this section we are presenting the summary of different virtual environments that are presented so forth. The summary are listed in Table 2.1

Table 2.1: Summary of the current most popular 3D Virtual Environments.

	SL	**WoW**	**PS-H**	**AW**	**Sims-O**
Subscription	Free	$20 to $200	Free	$6.95 per month	$9.95 per month
Public, private space	Both	Public	Both	Both	Both
Subscribers (million)	25[1]	12[2]	17[3]	3[4]	2[4]
Voice chat facility	Yes	Yes	Yes	Yes	No
Virtual currency	Linden Dollar	WoW gold	No	No	Simoleans
Intellectual property rights	Object Owners	Blizzard Inc.	Sony Co.	AW Inc.	EA Inc.
Platforms	PC, Mac, & Linux	PC & Mac	PlayStation 3	PC & Linux	PC
Scriptable content?	Yes	Yes	No	Yes	No
Open source available?	SnowGlobe	No	No	No	No

[1]as of August, 2011, [2]as of October, 2010, [3]as of December 2010, [4]as of January 2008

2.1.2 3D Avatar

According to Merriam Webster[1] an avatar is, "an electronic image that represents and is manipulated by a computer user". The term refers to a computerized representation of a human body in a computer-simulated virtual environment. Although 'avatar' usually refers to a virtual human representation, other representations of physical/living things like animals in the virtual environment are also called avatars. The term 'avatar' was first introduced by Joseph Romero & Chip Morningstar, for designing Lucas Film's role-playing game *Habitat* in 1985. According to Joseph Romero, a 3D avatar is a computer-generated 3D model which represents a virtual human, animal, or other living being, which can move and talk like the physical human, or animal, or living being that it represents.

[1]MerriamWebster, http://www.merriam-webster.com

Figure 2.4: A sample 3D avatar in Second Life Virtual Environment.

2.1.3 Linden Script

Linden Script (*LSL*)[1] is a scripting language developed by Linden Lab, which imparts the capability to create custom objects in Second Life. LSL script can be attached to any primitive object in the world, but not to an avatar. Avatars, however, can wear scripted objects. The syntax of LSL script is similar to the syntax of popular programming language Java. Like other scripting languages, LSL script has variables, functions, and events. One thing that makes LSL unique is its emphasis on *States* and *Events*. A door can be "open" or "closed" and a light can be "on" or "off". This state-based idea came from real world objects, since many real-world object behaviours can be modeled in the same way.

Each LSL script has at minimum one state, which is the scripts default state. An event can be thought of as a "Trigger". Events are predefined, and are triggered when two objects or avatars are colliding. For example, when an avatar touches an object, a "touch start" message is sent to the object, which causes the *touch_start()* event handler to begin executing. LSL script has over 500 functions available. Users can also define additional functions. Figure 2.5 depicts a sample LSL written in Second Life.

[1]Linden Script, http://wiki.secondlife.com/wiki/LSL_Portal

```
default
{
    touch_start(integer total_number)
    {
        llsay(0, "Hello, Avatar!");
    }
}
```

Figure 2.5: A sample script written in LSL. If the script will attach with any object and touch the other object then the *touch_start* function will be called and a message "Hello, Avatar!" will be displayed in the screen.

2.1.4 BVH File

Biovision Hierarchical (*BVH*) is a file format that contains motion capture data and is widely used in computer animation; it was developed by Biovision Inc, a motion capture company. Currently this file format is one of the most popular in the animation community because of its simple specifications. A *BVH* file ends with the extension '.bvh', which is nothing more than a text file and contains captured data from a moving skeletal system. There is no standard method for creating data for a *BVH* file; most studio houses that do their own commercial animation have their own type of motion capture technique. Second Life uses *BVH* file to animate avatars or objects. Users can upload their animation file (*.bvh*) to the Second Life server, on the provision that they have to pay L$10 for each animation upload. The step-by-step animation uploads mechanism is found in the Second Life Wiki [28].

Figure 2.6 demonstrates a general structure of a BVH file. According to this figure a *BVH* file has two major sections: "HIERARCHY" and "MOTION". The "HIERARCHY" section describes the joint-to-joint connections and offsets for the sampled motion data. The "MOTION" section describes the movement of these individual joints on a per-sample basis. According to this figure the "HIERARCHY" section is followed on the next line by the keyword "ROOT" and the name of the bone that is the root of the skeletal hierarchy. The "ROOT" keyword indicates the start of a new skeletal hierarchical structure. The remaining structure of the skeleton is defined in a recursive nature where each bone's definition, including any children, is encapsulated in curly braces, which is delimited on the previous line with the keyword JOINT (or ROOT in the case of the root bone) followed by the name of the bone.

```
HIERARCHY
ROOT Hips
{
   OFFSET[x_float y_float z_float]
   CHANNELS 6 Xposition Yposition Zposition Zrotation Xrotation Yrotation
   JOINT Chest {...}
   JOINT LeftUpLeg
   {
        OFFSET[x_float y_float z_float]
        CHANNELS 3 Zrotation Xrotation Yrotation
        End Site {OFFSET [x_float y_float z_float]}
   }
   JOINT RightUpLeg {...}
}

MOTION
Frames:    [number_of_time_samples_to_follow]
Frame Time:[0.033333]

[samp1_chan1_float][samp1_chan2_float]...[samp1_chanN_float]
[samp2_chan1_float][samp2_chan2_float]...[samp2_chanN_float]
  ...
[sampN_chan1_float][sampN_chan2_float]...[sampN_chanN_float]
```

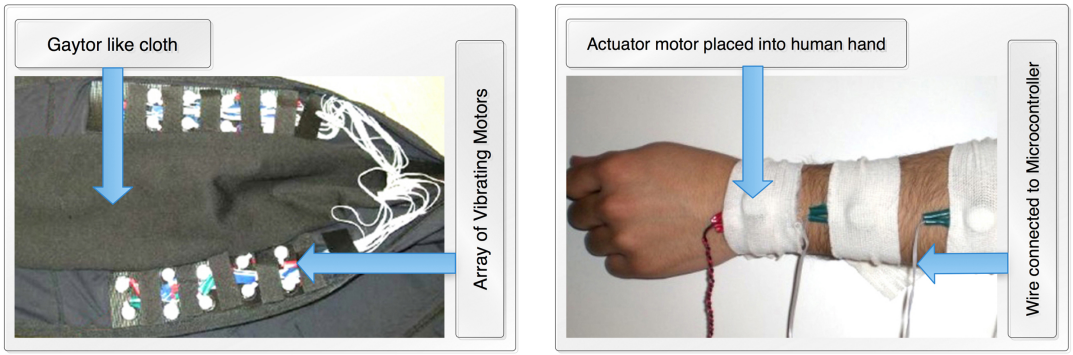
Figure 2.6: A sample *BVH* file that contain the main two parts "**HIERARCHY**" and "**MOTION**".

2.1.5 Vibrotactile Actuators

Vibrotactile actuators are an array of motors specially placed into a wearable surface like cloth or plastic to give a funneling illusion to the human body. Funneling illusion is one type of sensation that is generated by multiple vibrating motors. In our practical experience when two actuators have the same intensity, the illusionary sensation is normally created in the middle between them. But if the intensity levels are different, then the sensation is shifted or funnelled towards the higher-intensity actuators. Figure 2.7 demonstrates a set of vibrotactile actuators placed inside a cloth and on a human skin.

2.1.6 Haptic Jacket

Vibrotactile actuators communicate sound waves and create a funneling illusion when coming into physical contact with skin. A haptic jacket consists of an array of vibrotactile actuators that are placed to a particular portions of the jacket, and their patterned vibration can stimulate touch in the user's skin [3]. A series of small actuator motors are placed on a 2D plane within the jacket, in a certain manner. An AVR Micro-controller directs the vibration of these actuators. Figure 2.8 depicts the components of the jacket



(a) In a gaytor like cloth. (b) In human body.

Figure 2.7: Vibrotactile Actuators.

in more detail.

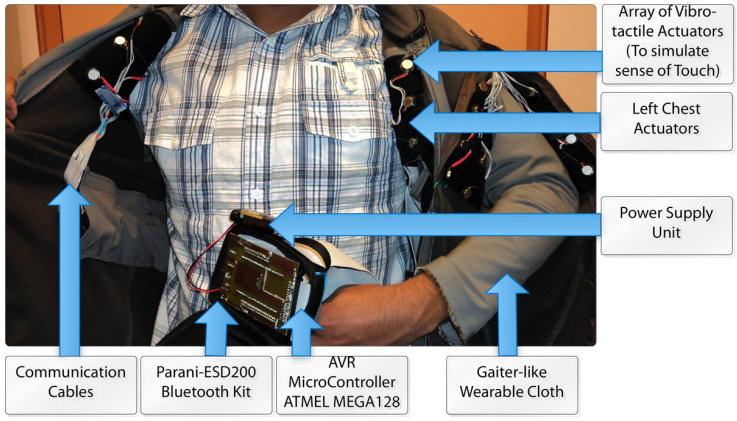


Figure 2.8: The Haptic jacket controller and its hardware components. Array of vibrotactile motors are placed in the gaiter like wearable cloth in order to wirelessly stimulate haptic interaction.

2.1.7 X10 Protocol

X10 is an old but popular power line communication technology, primarily used for controlling home electronic devices such as lights, fans, televisions, and music systems. The original X10 and X10 protocol was developed and designed by Pico Electronics of Scotland in 1975. Although several new technologies were invented after X10, this is still an open industry standard for communications among electronic devices because of its easy installation process and inexpensiveness. X10 uses the home power line to send

and receive signals between devices. The digital signal used to control the X10 modules (figure 2.9) is a piece of digital data that is encoded onto a 120 kHz carrier. Each unit of digital data consists of a home address represented by a letter between A to P, a unit address between 1 and 16, and a 4-bits command (table 2.2 shows some important commands). This simple data helps in controlling our home appliances: turning on/off a television, music system or light, increasing or decreasing the intensity level of a lighting or air conditioning system, controlling room temperature or humidity, and so on.

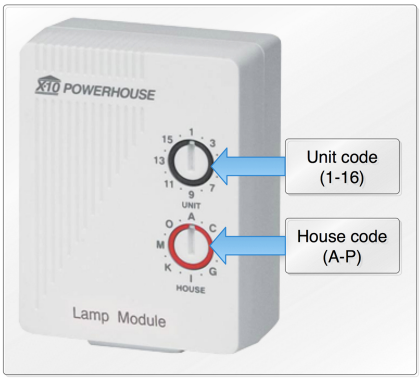


Figure 2.9: A 2-way X10 lamp module that demonstrating the 16 house codes and 16 unit codes that can control a total 16x16 different devices.

Table 2.2: Important X10 commands that are used to control the home appliances.

Digital Code	**Function**
0000	Turn off all the devices used in the home
0001	Switch on all the lighting devices in the home
0010	Switch on a specified device
0011	Switch off a specified device
0100	Decrease the intensity of a light
0101	Increase the intensity of a light

2.1.8 Web Service

According to the World Wide Web Consortium ($W3C$)[1], a web service is, "a software system designed to support interoperable machine-to-machine interaction over a network.

[1]World Wide Web Consortium, http://www.w3.org/2002/ws

It has an interface described in a machine-processable format. Other systems interact with the Web service in a manner prescribed by its description using *SOAP*-messages, typically conveyed using *HTTP* with an *XML* serialization in conjunction with other web-related standards." [47]

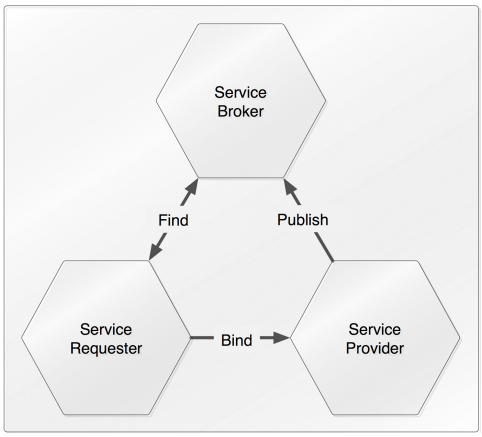


Figure 2.10: Basic architecture of a web service.

Figure 2.10 depicts basic web service architecture. According to this figure the architecture has three main components: service provider, requester, and broker and three simple operations: publish, find and bind. Any network module can fulfill any or all of the roles. The Web Service architecture provides several benefits, including: Promoting interoperability, reducing complexity by encapsulation and enabling interoperability of legacy applications.

2.2 Related Works

A few comparable inquiries have been carried out for bridging the gap between the virtual and the real world. These inquiries stand as a guideline for our work here. This section briefly discusses some of those systems and the basic concepts and architecture behind them.

2.2.1 IFeel_IM

Recently, Tsetserukou et. al. [45] [46] have attempted to analyze the text conversations in Second Life's chatting system. This system provides emotional haptic feedback to the users by using specially designed wearable hardware. The authors tried to bridge

the gap between mediated and face-to-face communications by providing a prototype system wherein different haptic devices give intimate emotional feedback during online communication.

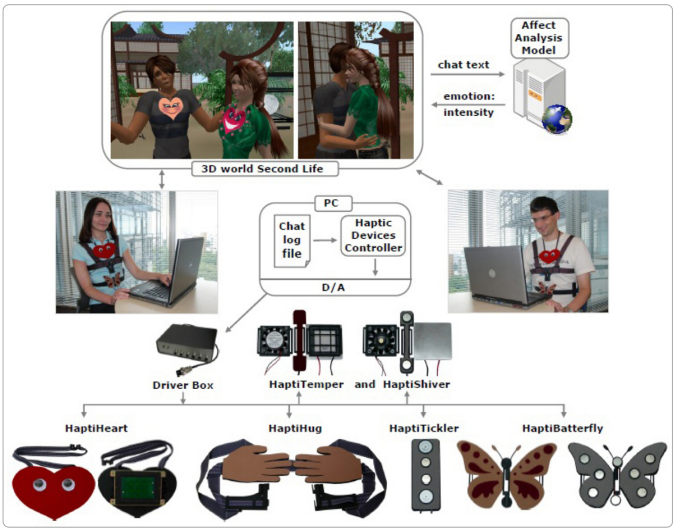
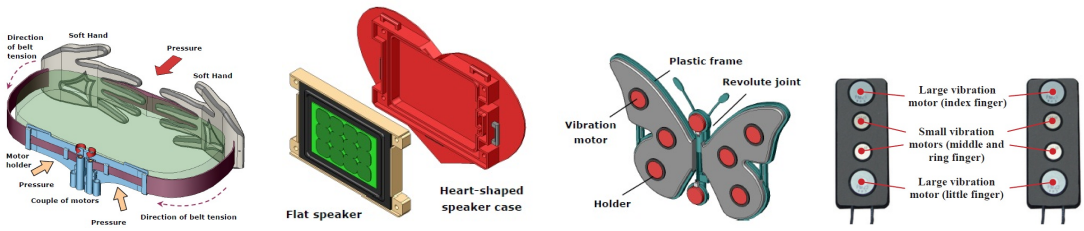


Figure 2.11: Basic architecture of the iFeel_IM system [45].

For intimate emotional feedback during online communication, they attached an object called *EmoHeart* to the Second Life avatar's chest. Then the system depicted in figure 2.11 listens to the communicated messages and sends all the captured messages to a web-based interface called an Affect Analysis Model (AAM) to visually reflect the behaviour. In the prototype system, EmoHeart is responsible for scanning the text-based chat messages and visualization of animation in Second Life. All the chat messages are collected in the permanent storage; later on, a haptic controller analyzes those stored messages in real time, and finally generates a signal for haptic feelings. In order to generate different haptic signals, the system has different types of hardware. Some of those haptic devices are presented in figure 2.12. The key feature of the designed haptic devices is the ability to generate strong physical features: 'anger', 'fear', 'sadness', and 'joy'. Some of those key features are as follows:

- The *HaptiHug* device has been developed to generate a hug pattern modelled only human-human interactions. A soft hand (figure 2.12a) is placed in the device in such a way that when any 'hug' command is received, the hand is then gently pressed to the chest part of a human body, to give the feeling of a warm human hug.



(a) HaptiHug (b) HaptiHeart (c) HaptiButterfly (d) HaptiTickler

Figure 2.12: Different haptic devices of iFeel_IM system (a) HaptiHug, (b) HaptiHeart, (c) HaptiButterfly, and (d) HaptiTickler for intimate emotional communication [46].

- The *HaptiHeart* (figure 2.12b) can create a heart-beat sensation with high fidelity.
- The *HaptiButterfly* (figure 2.12c) can generate joy or positive sensations by tickling the rib area of the interacting users.
- In order to create fear-like emotion or feelings, the *HaptiShiver* (figure 2.11) sends "shivers down/up a human body's spine" by means of a row of vibration motors (HaptiShiver), and "chills down/up a human body's spine" through both cold air flow from a DC fan and the cold side of the Peltier element (*HaptiTemper*). HaptiTemper is also intended for simulation of warmth to evoke either pleasant feeling or aggression.

While the various hardware designs of HaptiTickler, HaptiHug, HaptiButterfly and HaptiHeart are commendable, this approach does not seem to consider visual or pointer-based graphical interactions in the 3D environment, other than the text-based conversation system. For example, while using this system, it is impossible to interact with a particular part of an avatar that can generate different touch or hug feelings in connection with the respective body parts of the real user.

2.2.2 HugMe

HugMe (Synchronous Haptic Teleconferencing) [9] is a haptic jacket based rendering of touch into a conventional teleconferencing system to provide haptic interactions to the remote users. This approach uses marker-tracking techniques to specify touchable parts of the user's body. The markers are further tracked using a dedicated camera. The system employs an expensive 3D camera in order to automatically create 3D touchable surfaces of the user. Figure 2.13 demonstrates the architecture of the HugMe system.

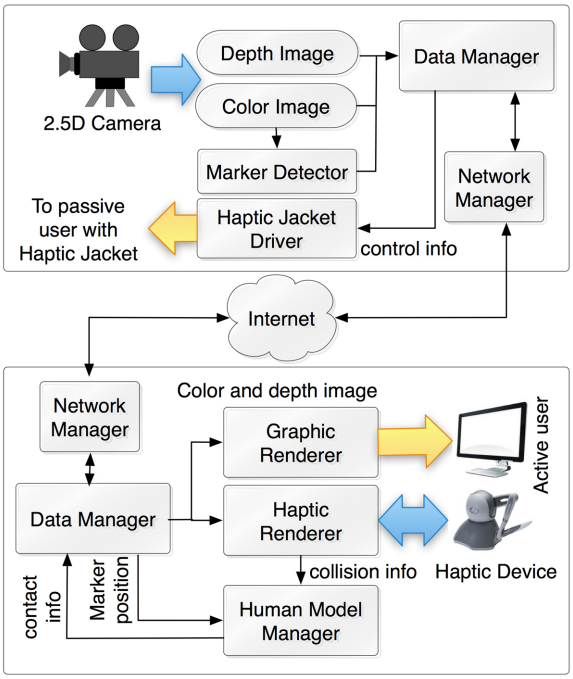


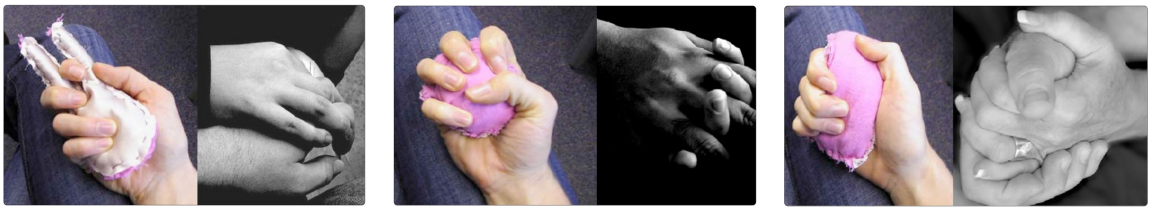
Figure 2.13: HugMe: Synchronous haptic teleconferencing system architecture [9].

By using this system, two remote Internet users could see as well as touch each other through a 2.5D captured interface and a wearable haptic jacket that is composed of an array of vibrating motors. For capturing 2.5D scenes, the system uses a depth camera, called ZCam. The captured images are termed 2.5D, in the sense that the depth image has incomplete 3D geometrical information. After capturing the images, the system incorporates the 3D graphic transformation to prepare depth images and to calculate the 3D interaction position for the actuator motors. An avatar which represents the passive user is used to map the interaction point in the depth image with the user's skin coordinates. Moreover, in order to track the movement of the avatar, a marker detector called ARToolKit [29] is used.

2.2.3 Holding hands over a distance

O'Brien et. al. [34] investigated an approach relating to intimate communication for couples. In this approach, a person could virtually hold hands with another, by using their proposed probe (a stress ball with a chip for logging data) to share tactile expe-riences with his or her partner's hand. They placed a small microchip inside the ball. When the ball is squeezed by a user, the system sends vibrotactile data to the other ball

that his or her partner is holding.



(a) Wrapping fingers (b) Intertwining fingers (c) Clasping hands

Figure 2.14: Three of the more common hand-holding styles into three form prototypes, using cotton stuffing, foam rods, and velvety fabrics [34].

According to O'Brien it is an ancient rule that when couples are apart, they try to communicate with each other through existing technologies such as telephone, email, SMS etc. Yet, other important means of communication, such as holding hands, can only happen when couples are co-located. Their system suggests a technology probe by which remote young adult couples can touch each others hands. They modeled three of the more common hand-holding styles into three form prototypes, using cotton stuffing, foam rods, and velvety fabrics (Figure 2.14). From their observations, they found out that the communicating couples like to hide their technology probe from the public; and in most cases, they forgot the probe at home when going outside. The probe design played a decisive role in the probe's effectiveness in collecting data. When the technology probe design is simple, the probe's effectiveness increases significantly, and it is more appealing, which makes the couple more likely to use.

2.2.4 Slow Messaging

Remote touch and intimacy is a crucial element of social life, and many interactive technologies are designed for emotional connection. Couples use a variety of means to maintain an emotional connection. The work proposed by Slow Messaging [23] posits that for couples in long-distance relationships, interactive communication technologies may be a primary means of exchanging emotions. Such couples often customize and/or create their own unique way of communicating to meet their needs. Flowers and love letters have long been used to express emotions between close people. Now-a-days, people use numerous techniques and technologies during the physical absence of their partner to maintain an emotional connection and to create a sense of presence-in-absence. SMS,

webcams, emails, instant messaging, and blogs are examples of such technologies used as mediators of human interaction for people living at a distance, and for romantic couples in distance relationships.

2.2.5 Him

In Instant Messaging (IM), Rovers and van Essen [37] have provided a detailed study of the usage of "hapticons" which essentially are vibrotactile icons representing smilies. They combine traditional textual messages with haptic effects and hapticons. Moreover, they incorporated six vibrotactile patterns that represent six associated smilies. These smilies could be triggered using mouse- or keyboard-based interactions. Like other ordinary instant messaging systems, HIM can provide features such as listing who is online and sending text messages. The architecture of HIM, as shown in Figure 2.15, has two main components; one is the server component and the other is the client component.

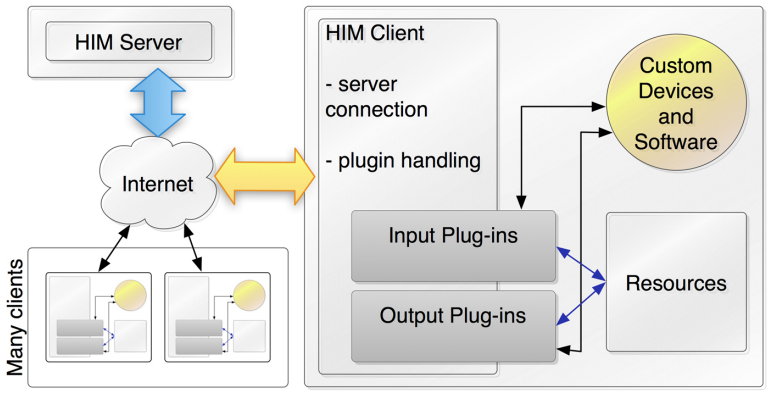


Figure 2.15: Haptic Instant Messaging (HIM) Architecture [37].

The server component is responsible for preparing the list of currently-online users and preparing a session (and the necessary resource allocation) for a chatting party or group. All messages are sent via the server. On the other hand, the client component runs on the users PC, and connects with the server to send or receive messages and to fetch the online user list to display on the user screen. Moreover, the client component can handle custom devices through plug-ins (Figure 2.15).

2.2.6 3D Virtual Smart Home User Interface

One of the supporting studies related to object-based smart home control using a 3D virtual environment is L. Borodulkin [6]. As the prices of smart home automation devices are decreasing, the need to control those devices with an efficient user interface is ever greater. L. Borodulkin et. al. [6] shows the important value of a 3D user interface over a 2D interface. Moreover, the author shows how a 3D user interface can improve the intuitiveness and adaptivity of smart home control, as compared to an ordinary 2D user interface. The author also shows that with the 3D user interface user requires less training and learning over 2D user interface because 3D user interface is more realistic and more like our physical world. With a virtual reality approach, the user fully benefits from the advantages of the 3D view design: instead of abstract objects labelled "room 1", "light left" or "heater 2" and a hierarchy of windows, icons and data files, a realistic view of the house with its structural components such as rooms, stairs, and windows is given to the user.

2.2.7 Design and Evaluation of Smart Home User Interface

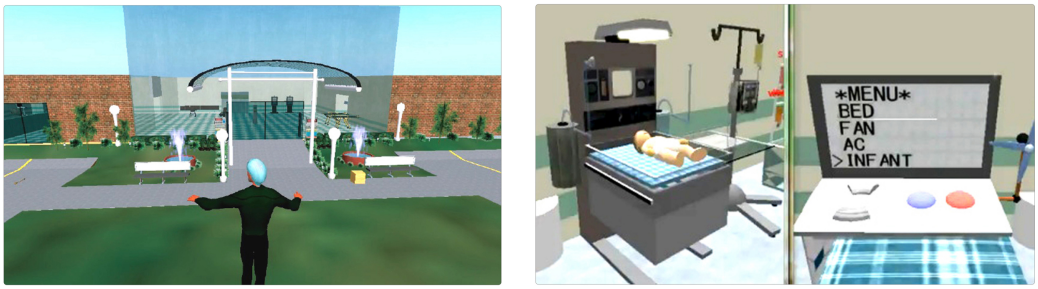
A user interface is not only important for efficient control of the system but also depends on the ages and intelligence of users. The work presented by Bin Zhang et. al. [53] showed how those factors affect the design of user interfaces, specially for the smart environment. Those factors are very important and have a high impact on users' cognitive tasks inside a home. Based on the user study, they found out that for skill-based tasks, users would obtain best performance with low-intelligence-level interfaces. On the other hand, for rule-based tasks, users would obtain best performance with high-intelligence user interfaces.

2.2.8 Modeling Health Care Logistics in a Virtual World

This prototype system was developed by the University of Arkansas [44]. The system, called Virtual Razorback "Hogspital", is installed in Second Life on "University of Arkansas Island" [1] (see Figure 2.16). The motivation of this project was to use a virtual world as a platform for modeling certain aspects of health care and a hospital's supply chain-from receiving goods to patient care. Just like a real-world hospital, the virtual hospital has virtual furniture and equipment, as well as toilets, sinks, showers, chairs,

[1] University of Arkansas SL project, http://vw.ddns.uark.edu

classrooms, patient rooms, beds, etc. It also has virtual patients, doctors and nurses with their correct uniforms. Moreover, the hospital has functioning units including virtual pill bottles, medicine supplies, rubber gloves, X-ray units, virtual RFID readers, blood pressure monitors, and CAT scan machines.



(a) Hospital main entrance (b) Health care remote control

Figure 2.16: Virtual Razorback "Hogspital" on "University of Arkansas" island in Second Life [44].

2.3 Summary

In this chapter we broadly presented the background study and some popular works that are related to this thesis. At the outset, we presented a comparative study of 3D virtual environments and related topics including 3D avatars, Linden script, BVH file and more. We conclude that there is a need for a system that dynamically update real world objects based on users' interactions in the virtual world and vice versa. Hence we try to propose such a system in the following chapter.

Chapter 3

System Design

This chapter illustrates the overall architecture of a system that facilitates real-to-virtual and virtual to real communication, along with detailed descriptions of the main features of the developed prototype tool. At first, the input controller and its different components are presented. Then the functional aspects of the prototype, including animation rendering and physical object rendering are discussed in detail. Furthermore, the secure communication channel and its working process is presented in a separate section.

3.1 Overall System Architecture

This section presents the different components of the system and describes their functions. Firstly, section 3.1.1 outlines the overall architecture of the system. Further, section 3.1.2 presents the Use Case model for the effective interaction of participating users. Finally, section 3.1.3 details the features of the prototype system.

3.1.1 Overview

We propose architecture to minimize the gap between the virtual and the real environment. The main idea behind our approach is to dynamically communicate with virtual objects and to render the events to give a suitable real-world signal, which can be used to generate feelings or to control physical spaces such as homes and offices. Figure 3.1 shows a schematic view of the overall system. As depicted in this figure, the whole system is controlled by a input controller and an interaction controller. The system uses the already-existing communication channels of the 3D virtual world, and so for security

reasons, it uses its own secure communication channel to send or receive messages. The various components of this figure will be described in the following sections.

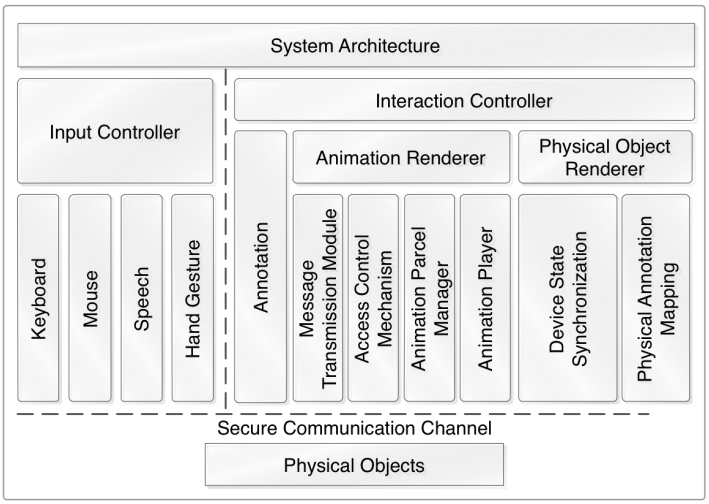


Figure 3.1: Schematic diagram of the proposed system.

3.1.2 Use Case Model

We will consider high level overview of the system that focuses on the specific features required to bring effective interaction to the participating users and devices. Figure 3.2 illustrate the use-case diagram to summarize the core requirements of the prototype system. Generally two basic actors will comprise the interaction process. Below we have discussed the different core Use Case of the model.

- **User subscription:** Before using the virtual world user have to subscribe to the system. Usually most of the virtual world has the subscription feature through their website. Each subscribed user have a 3D avatar and a home location where the avatar landed (usually called *teleport*) after successful login to the system.
- **Personalization:** A user can create or modify his/her profile. The user updates his/her preference choices (or personal group list like friend list, family list etc) which later is used to build the personalized communication between the virtual avatar. Moreover, user can add customizable animation, objects into his/her inventory list.

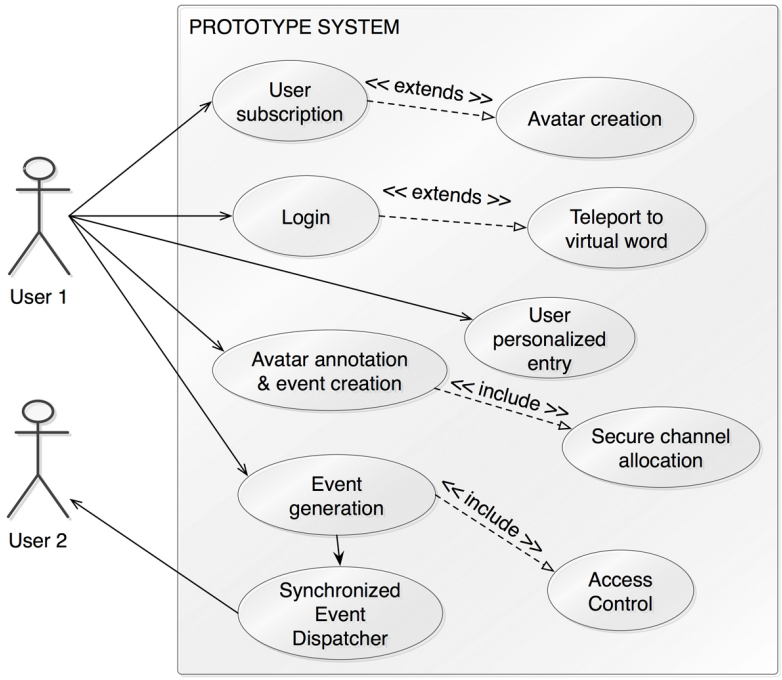


Figure 3.2: Use Case diagram to summarize the core requirements of the prototype system.

- **Annotation and event define:** A 3D object can be annotated and attached XML based rules and can be changed its usual behaviour. When any event triggered then the object read its associated XML rules and act accordingly.
- **Event generate and channel allocation:** For each rule that is associated with an object (avatar or 3D objects) can generate other events and can allocate channel for secure communication. Whenever any secure channel allocated then the communicating parties can send or receive the haptic or device command to control the real objects or simply generate the haptic vibration in the actuators.
- **Synchronized event dispatcher:** One of the biggest challenges of the communication between real and virtual is to generate synchronized animation with the real object. The synchronized event dispatcher control the animation played in the virtual environment with the real life object behaviour by monitoring and managing the generated events. Usually it receives the event command and stores it into an event queue and generates the synchronized behaviour.

3.1.3 System Features

The prototype tool presents an implementation in accordance to the presented architecture and offers highlighted features:

- Can easily incorporate annotation rules and XML with an object that able to communicate with the other module of the system without restarting the virtual environment.
- Using software add-on architecture to design the prototype system, so that it can attach or remove to/from the existing virtual environments.
- Haptic interaction opportunity between the real users and their respective virtual avatars through a 3D graphical user interface using speech, mouse, text and gesture based interaction modalities.
- Introduce touch, hug and tickle haptic features for the Second Life users through chat and GUI (Graphical User Interface) interactions.
- 3D annotation mechanism for the Second Life avatar so that user dependent interpersonal haptic and animation interactions become possible.
- Creates and maintain per-user or per-group annotation rules to receive different touch, hug feelings from different persons or groups.

3.2 Input Controller

The Interaction Controller works as a core service and takes action according to the user inputs from the input Controller (Figure 3.3). The input controller enables the usage of keyboard, mouse, speech and gesture based inputs from the user. For example, a user representing a female avatar can point her mouse on a male avatar and produce a click event using the mouse. The input controller detects if the annotated body parts of the male avatar has received any GUI commands and sends the avatar body ID and type of action performed to the Interaction Controller. If the user clicks on the annotated object then the annotated XML file will be executed by using the existing scripting feature of the virtual environment. In our prototype the hug command is issued by using the speech, keyboard and gesture based interaction inputs. The GUI commands that were used in the various interaction inputs are discussed in the following sections.

Keyboard

While processing the keyboard (text) based inputs from the requester the controller analyzes the text messages and later on sent to the jacket owner or the respective devices. The text message based commands have certain preamble before the commands. As our system depend on the interaction in the particular region of an avatar or virtual object so using text based interaction through keyboard is not sufficient to generate all the command messages. Therefore, the interaction controller easily distinguishes the haptic and object commands that are issued based on the text inputs. The text command forms are HUG <*username*>, TOUCH <*username bodyparts*> and TICKLE <*username bodyparts*>, where *bodyparts* = {*leftChest, rightChest, stomach, leftShoulder, rightShoulder, leftBackShoulder, leftRightShoulder, leftArm, rightArm, neck*}. For device control the interaction command should be STATE < *level** *deviceAddress*>, where STATE is the new state of the device like ON/OFF or OPEN/CLOSE, *level** is a optional input between 0 to 15, this input specifically designed those devices which have multiple level of states. For example temperature or humidity sensors. By STATE we can on or off a temperature sensor and by *level** we can set the temperature level. The last input is *deviceAddress*, the actual device address that we want to control.

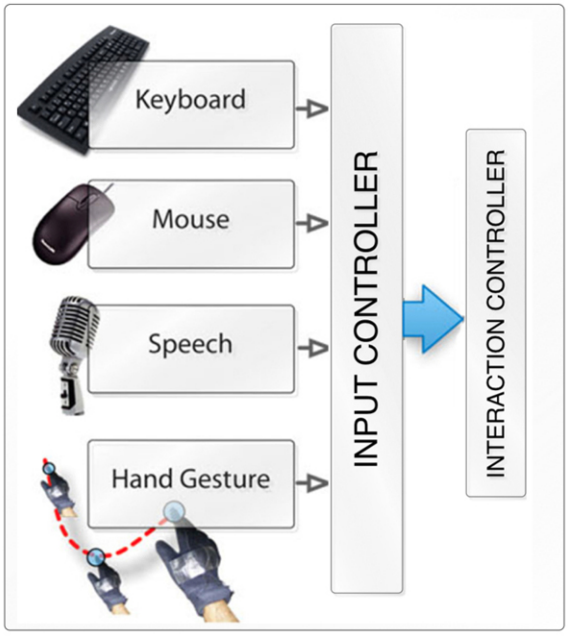


Figure 3.3: Input controller that take input from different input devices and send the processed messages to the interaction controller.

Mouse

It is flexible to provide touch and tickle commands using a mouse. For each mouse click at the annotated body parts a touch command is issued. When the mouse click happens on the stomach and neck area of the virtual avatar a tickle command is captured. In order to provide hug command the user has to right click on the annotated part of the respective avatar and have to choose hug option. This will issue a hug command automatically. Provided that the other user have to approve the hug requester message. If the hug is first time then a pop-up message shows in the receiver user machine, if he/she will approve the hug then a hug will be initiated, at the mean time the permission will be stored for future interaction. However, the user can remove the permission at any time. In case of controlling the physical devices through mouse the object owner can click the annotated part and this will initiate a command based on the current state and the level of the virtual object.

Speech

We incorporated the existing speech based interaction methodology in virtual environment [14]. We processed the speech based haptic commands from the user. The touch and tickle input commands are similar to that of keyboard interaction, where the user speaks out the type of interaction (touch, tickle) followed by the body part names. In order to issue hug input command, the user simply speaks out hug and the nearest user is issued a hug command. In the same way in order to control the other objects we need to say the device state followed by the device address. In our system there is a limitation for speech based interaction. As user name recognition was not attempted in our approach so we just recognize only some predefined name tags such as "tickle", "hug", "light 1", "fan 2" etc.

Hand Gesture

In the system we added the facility to control the device or avatar based interaction by using existing novel motion path based gesture interaction system[35]. This system allows the user to define a drawing symbol that can be associated with particular command. We tailored the motion path based gesture interaction approach by introducing four main drawing symbols e.g. h, T, k, S representing hug, touch, tickle and device control commands respectively. For the avatar body parts we associated the following gesture commands, $bodyparts = \{leftChest(L,C),\ rightChest\ (\Gamma,C),\ stomach\ (S),$

leftShoulder(L,S), rightShoulder(Γ,S), leftBackShoulder (L,b) , leftRightShoulder (Γ,b), leftArm (L,m) , rightArm (Γ,m), neck(n)}. The gesture drawing symbols were chosen based on their selection accuracy. For the device control when the user draw the symbol S then the system will understand that user wants to send a device control command, and wait a certain amount of time for the next input. When the system receives the device address and the state symbol then it will process the whole command. For example when the user draws a symbol or gesture like "OA1" which indicates that the user wants to turn on the device that has address "A1". Similarly if the user wants to increase the room temperature then they can draw "I5A2", which means increase the temperature by 5^oC that have address "A2".

3.3 Interaction Controller

In representing the interaction controller, the engine of the system, we will introduce the object annotation procedure in section 3.3.1. The annotation provides personalized animation and physical object control options. In order to generate more intuitive real world events we introduce multi-region based annotation process that will be described in section 3.3.1. Moreover, section 3.3.2 presents the animation rendering process and its different components. Finally in section 3.3.3 presents the physical object rendering process in details. Overall the interaction controller employs the following logic as defined in Algorithm 1 in order to bring the virtual events into the real world users through avatar interaction.

Algorithm 1: Real Virtual Interaction

Require: Animation BVH file, annotated XML and device command data.

Ensure: The hardware interface is synced with the avatar/object animation.

1. Decode the device command data to obtain animation and device pointers.;
2. Load animation data.;
3. Prepare serialization of the web-service data.;
4. Obtain device state and prepare the Parani Bluetooth kit or X10 module.;
5. Apply parallel data transmission.;

Figure 3.4 demonstrating the interaction controller and the other communicating modules. The interaction controller receives the interaction data from the input controller. The interaction data further processed by the animation renderer and physical object renderer modules. Finally, the processed data send to the secure communica-

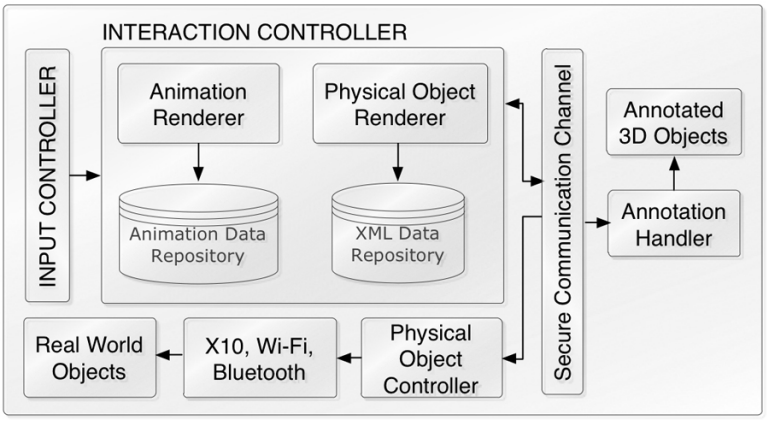


Figure 3.4: Interaction controller communicating the different modules of the system. It takes signals from the input controller and send the signals to the secure communication channel.

tion channel. The working process of different modules of the interaction controller are elaborated in the following sections.

3.3.1 Annotation

Annotation mechanism is one of the primary parts of our proposed system. We can create *XML* based rule and can attach the rules to the virtual 3D objects surface (i.e. avatars or objects). Later on the object will behave based on the attached rules. Although the annotation approach of virtual avatar and the other virtual object is almost similar but for clarity reason we will present the annotation process of avatar and the other virtual objects into two different sections.

Avatar Annotation

In our system we annotated visible body parts of the 3D virtual avatar and specified the corresponding physical haptic actuators to render the haptic feedback. For each haptic signal we also annotated the avatar animation. Figure 3.5 depicts the geometric based avatar annotation scheme of Second Life. We attached LSL scripts [27] in each of the annotated parts of the avatar that contain the haptic commands as well as the identification number of the animation sequences. For example, we annotated the 3D male avatar's left arm and specified particular vibrotactile actuator stimulation for it.

Further, we specified the interacting animations for both the participating male and

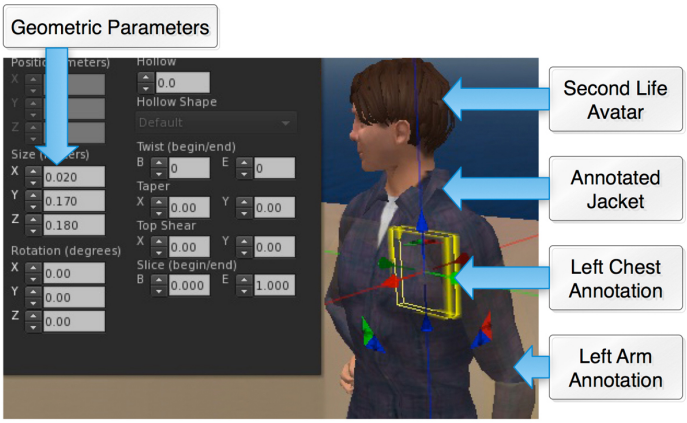


Figure 3.5: The flexible avatar annotation scheme allows the user to annotate any part of the virtual avatar body with haptic and animation properties. When interacted by the other party, the user receives those haptic rendering on his/her haptic jacket and views the animation rendering on the screen.

female virtual avatars. Afterwards, when the user representing the female avatar issues a GUI (Graphical User Interface) interaction command to the male avatar arm then the annotated haptic stimulation is rendered at the real male user's arm through the haptic jacket. For intimate interactions such as a hug, we employed group based annotation scheme. As evident, hugging with parents is different to that with a friend. Hence, we needed separate animation and haptic rendering for each type of hugs, touch etc interactions. We created groups and incorporated group based annotation scheme of the 3D avatar. For each group we created different avatar animation and haptic rendering options. By using the script based dialog interface any interacting contacts were then assigned to a group (default is formal). We provided four different groups namely family, friend, lovers, and formal. This group based haptic interaction in virtual environment further assisted the user to personalize his/her experience.

Object Annotation

Like the avatar annotation that we described in the previous section the object annotation process is same but in this case we annotate the object rather than avatar body parts. Although the avatar will control the behaviour of the object but object will be the primary place for annotation. In our system we annotated Second Life 3D objects and specified the corresponding physical device addresses. Every object in the real life has unique id by which smart home services can control it.

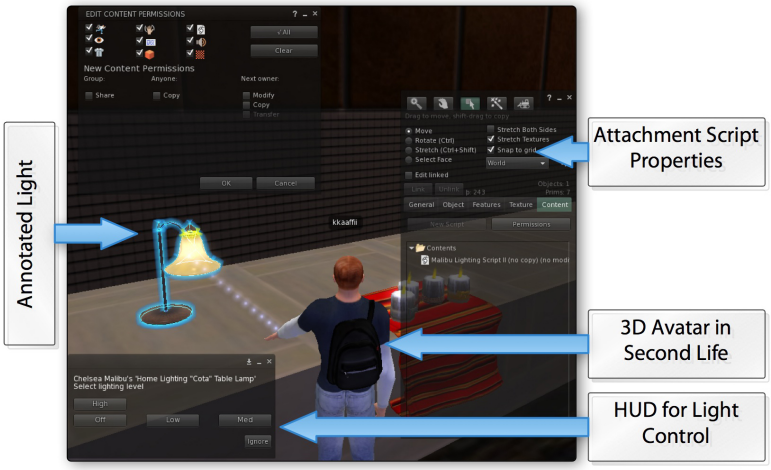


Figure 3.6: The flexible object annotation scheme allows the user to annotate object and to set animation properties. When interacted by the home users or automated triggering, the users see animation rendering on the screen.

```
<?xml version="1.0"encoding="utf-8"?>
<AnnotationRules>
   <animationModules >
     <animationModule id="doorOpen1">
        <UUID>6b61c8r8-6878-0h87-12hg-j1hj3h4us8g9</UUID>
        <animationBVH>doorOpen.bvh</animationBVH>
        <animationLooped>NO</animationLooped>
        <animationSpeed>30</animationSpeed>
        <animationSpeed>LOW</animationSpeed>
        <animationScaleTo>1</animationScaleTo>
     </animationModule>
     <animationModule id="doorClose1">
        <UUID>6f51c8r8-4555-0h87-89dw-a02j4b49sk2b</UUID>
        <animationBVH>doorClose.bvh</animationBVH>
        <animationLooped>NO</animationLooped>
        <animationSpeed>30</animationSpeed>
        <animationSpeed>LOW</animationSpeed>
        <animationScaleTo>1</animationScaleTo>
     </animationModule></animationModules>
   <Appliances>
     <Appliance name="Main Door 1">
     <ApplianceType>Door</ApplianceType>
     <DeviceType>X10</DeviceType>
     <PhysicalAddress>A1</PhysicalAddress>
   </Appliance>
</Appliances>
</AnnotationRules>
```

Figure 3.7: An overview of the target object specific interaction rules stored (and could be share) in an *XML* file.

The 3D objects are annotated using the script annotation mechanism that is demonstrated in Figure 3.6, where the 3D lamp is annotated with the help of a separate annotation file. The annotation file is an *XML* file that specifies the animation unique identifier (*UUID*), the animation file (*BVH*), the speed of animation, duration of animation etc. This files also species the physical device specific data like device address, name, type etc. In Figure 3.7 depicting a sample *XML* specification that demonstrates a door open close scenario where two animations "doorOpen1" and "doorClose1" are specified for open/close the real door that is connected with an $X10$ appliance module. Here the device address is also specified which is "A1" in this case. In the annotation phase we specify the LSL scripts [32] using the Second Life viewer obtains control of the objects and performs read/write operation into its communication channel. A sample LSL script for the door example is depicted in Figure 3.8.

```
state_entry()
{
    //Request animation permission to Second Life
    llRequestPermissions(llGetOwner(),PERMISSION_TRIGGER_ANIMATION);
    llSetTimerEvent(1.0);

    //Specify communication channel
    llListen(&iChannel,"",llGetOwner(),"");
}

start_animation(Object annotatedXML, string id)
{
    //load animation object from attached XML data
    loadAnimationModuleFromXML(id, &animationObj);

    //Render animation
    llStartAnimation(llList2String(this, animationObj));

    //popilate the device object list from attached XML data
    applianceObjList = getApplianceModuleFromXML();

    //write data to the channel to send the message
    Write_channel(llList2String(null, applianceObjList));
}
```

Figure 3.8: Code snipped of SL Script to load the XML data to send or receive messages into the communication channel.

Multi-region Annotation

In order to get refined response and to provide high flexibility to control the physical objects our annotation scheme introduces a generic interface. This is especially important for those devices in which different parts of a virtual device need different annotation

rules. For example, a table fan have different switch for speed control whereas a lamp has different illumination intensity levels or a temperature sensor have different temperature levels. In order to provide this type of facilities we developed a suitable annotation mechanism that allows annotation into different parts of the 3D virtual object. In this manner we specify rules in the XML data that we need to attach with specific region of the object. This type of multi region annotation scheme is depicted in Figure 3.9.

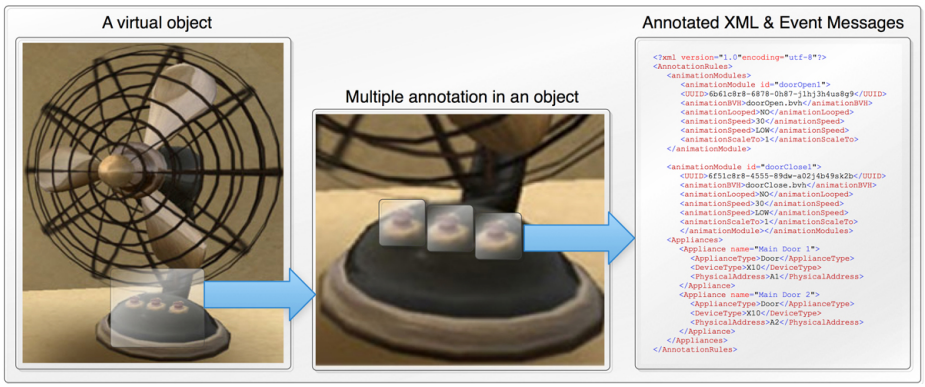


Figure 3.9: Multi-region annotation in a 3D fan and specified a physical lamp address which is connected to a X10 or WiFi.

3.3.2 Animation Renderer

In this section we present the animation rendering process and the associated modules. We start with the working process of the message transmission mechanism between the real and virtual world. This is followed by the security and authenticity process for the interaction and the animation sharing mechanism. Finally, the working process of the animation parcel manager is detailed.

Message Transmission Module

Message Transmission Module (*MT*) is responsible to bridge the virtual environment and our add-on. It receives all the output messages from the Nearby Interaction Event Handler (*NIEH*) in an encrypted XML [48] format. In virtual environment all messages are valid within a particular area, which is dependent on the avatars virtual 3D location. This area normally is defined as 10 to 30 square meters centered on the user's virtual location. All the messages generated within this area handled by *NIEH*. Normally in every virtual environment a message transmission module captures all the messages and

then Nearby Interaction Event Handler filter those messages which are valid for the region and later on send the filtered messages to the network. The MT module monitors the messages that are sending or received by the *NIEH*. Later on this module decrypts those received messages and transmits them further to the communication channel towards the Access Control Manager.

Access Control Mechanism

In the system user have the facility to incorporate user profile specific access control mechanism in order to provide the participating users the means of authenticating and personalizing their interactions. This type of access control based authentication is important for virtual environment based interaction because the environment is a shared place, any avatar can be initiating the request. If the user wants to take haptic feelings only from a few peoples then access control based permission system is effective. For example if user A issues a hug command to user B then the animation and haptic rendering take place only if user B acknowledges the permission. A permission window is shown at user B's window for this purpose, where the interaction could be accepted or rejected. We used Second Life message notification and graphical user interface to display the permission window in which the user is already adapted. In Second Life each user is associated with a string based identification number. Message originated from a user's computer bears that identification number as a preamble to that message. Hence, in order to provide access control we compared the identification number with the list of contacts of the user and decided accordingly. In order to deliver user specific haptic feedbacks to the user we used the group annotation. In any haptic interaction, the originator user information is mapped to obtain the group of the user. This phenomenon is depicted in Figure 3.10. In this approach the physical object renderer uses the group specific avatar animation and haptic rendering data in order to deliver customized interactions to the users.

Animation Parcel Manager

An Animation Parcel Manager (APM) is a container that contains the BVH file and the associated resources to play the animation. In our system, we do not store the animation and other resource file in the virtual environment repository. Instead, we suggest storing the resource file in our own repository; when the interaction happens, then the associated resources are sent to the virtual environment to process the animation

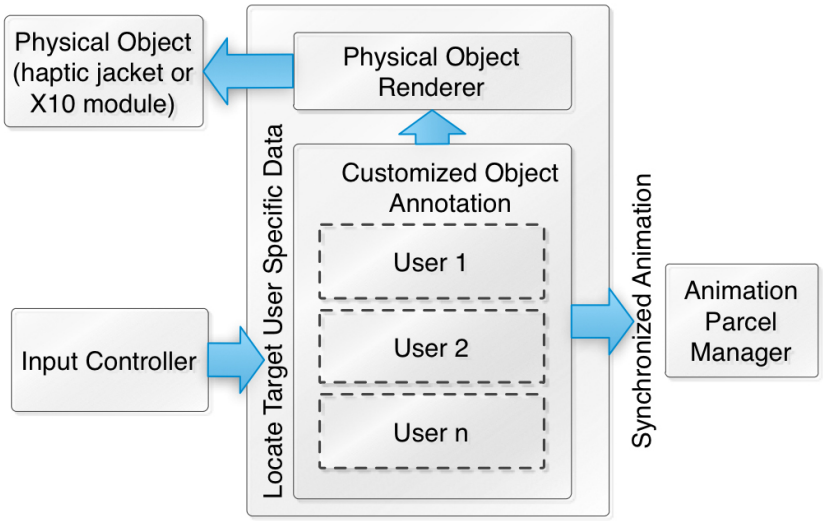


Figure 3.10: User dependent interaction access design.

and other information. An *APM* combines the entire resource file in order to make communication easier.

3.3.3 Physical Object Renderer

This section presents the physical object rendering process. First, we outlines the haptic rendering process. Later on, we describes the device state synchronization process.

Haptic Renderer

The haptic jacket provides the funnelling illusion based touch haptic feedback. We leveraged the touch feature to create hug and tickle based haptic feedbacks. We made careful observation about the real life hug and noticed that when two people hug each other both feel a gradual touch feeling in some specific body parts. In a formal hug a user receives touch feedbacks at the chest area and at the back shoulder area. Similarly, during our observations we noticed that in a tickle most users react to the random touch at the stomach area, at the underarm area and sometimes at the neck area. Using these empirical parameters we constructed touch, hug and tickle haptic feedbacks as the following:

- According to the virtual annotation the haptic touch sensation is delivered by incorporating the funnelling illusion into the haptic jacket to stimulate real touch at

the real user. When one person touches another person then both the participating users receives touch feelings.

- In order to create hug type haptic feedbacks for the participating users we systematically increased the jacket's *leftChest*, *rightChest*, *neck*, *leftBackShoulder*, and *rightBackShoulder* motors intensity levels to produce the funnelling illusion. The systematic control of the actuator intensity levels creates the touch effect in those areas and offers a hug type haptic stimulation. The lover type hug is different to that of the formal hug. In addition to the areas defined above we decided to add haptic touch stimulation in the stomach area to emulate the joy emotion [13] [45]. Hence, by following the laws of funnelling illusion we activated the arrays of vibration motors attached to the abdomen area of a person.

- As described earlier, from our empirical study the tickle haptic feedback is evoked by incorporating random and unpredictable touch at the stomach area, at the underarm area and at the neck area provided that a GUI interaction at those virtual body places were performed.

Device State Synchronization

In order to render synchronized animation in the virtual environment as per the changes in state of the real device, we incorporated a state-based event communication mechanism. When a user clicked or touched a 3D object in the virtual environment, the system received the event message which was then saved into an event queue. Further, the system processed the event queue sequentially and transmitted the message respectively to the receiver (see figure 3.11). At this stage the physical device became activated to handle the input messages. In case the device failed to initiate its services, an error message was displayed; otherwise it sent a Ready or Acknowledgment (*Ack*) message. In the next step the system sent the actual message, such as "open door", "close door", "increase light intensity to high", "decrease sound system volume to low" etc. The smart home service received the message and sent the command messages to the specific X10 or Wi-Fi module. The add-on also sent a message to the VE communication channel to play a synchronized animation that gave a realistic intuition to the user. When the physical object tried to close the device or went to reset its state, the smart home service received this updated state and sent the confirmation to the add-on to stop/resume the animation, as the physical device had already changed its state. This state-based interaction process works as illustrated in algorithm 2.

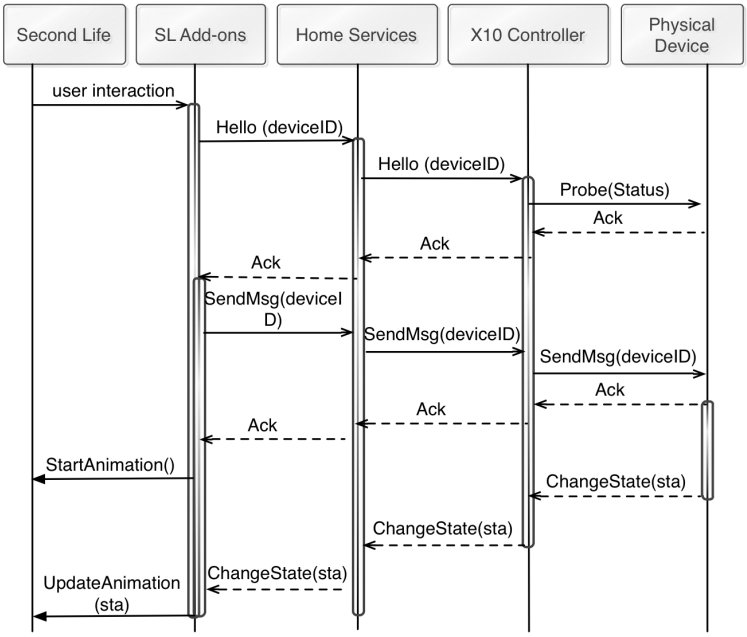


Figure 3.11: Interaction diagram to control a physical device by using our proposed system.

Algorithm 2: *StateSynchronize*(D_n, α_n, β_n, φ, υ)

```
/* D_n is the device name, α_n is the new state and β_n is the new
   level of the device which state should be synchronized.  φ is the
   animation id and υ is the avatar or virtual object UUID.        */
begin
    d ← getDeviceIdByName(D_n);
    α_c ← getState(d), β_c ← getLevel(d);
    λ is the maximum wait time;
    if α_c ∈ ERROR then        /* Device is dead so stop the animation */
        resetAnimation(φ, υ, 0)
    else
        if α_c ∈ READY and getAnimationStatus(φ, υ) ∈ READY then
        /* both parties are ready to work */
            resetAnimation(φ, υ, 0);
            setState(α_n, β_n, d);
            return;
        if α_c ∈ READY and getAnimationStatus(φ, υ) ∈ PLAYING then
        /* Device is ready but the animation is not ready then wait
        for the animation to end */
            while getAnimationStatus(φ, υ) ∈ PLAYING do
                if TOTALWAIT > λ then
                    return;
                wait for a specific amount of time for recheck;
            reset both the device and the animation;
    return;
end
```

3.4 Secure Communication Channel

The primary responsibilities of the Secure Communication Channel are to secure the messages passed from a sender to a receiver. A communication channel is a dedicated connection between two parties. In our proposed approach we used a full duplex socket based communication channel. As the connection is full-duplex both the sender avatar and the receiver avatar can send data at the same time. In order to ensure security before sending any messages to the channel it automatically encrypts the data that

passes through it and decrypts the data in the recipient section automatically. In each data transaction, sender and receiver section and other originators are verified before their messages are forwarded.

3.5 Summary

In this chapter we broadly presented the overall system architecture along with detailed descriptions of the main features of the developed prototype tool. At the outset, we presented the functional aspects of the prototype, including animation rendering and physical object rendering precess in detail.

Chapter 4

Implementation

The prototype system has been implemented using the design concepts described in the previous chapter. This chapter presents the actual implementation process of the prototype. Additionally, we will provide implementation details of the software architecture, platform and deployment issues, and present some sample interfaces.

4.1 System Requirements

Table 4.1: Different components of the prototype system.

Software	**Version**	**Description**	**Purpose**	**Publisher**
Snowglobe	1.4	Second Life Viewer	Virtual world client	Linden Lab
CMake	2.8.1	Cross-platform build system	Viewer build	Kitware Inc
OpenCV	2.1	Computer Vision Library	Gesture recognition	Intel
SAPI	5.1	Speech SDK	Voice interaction	Microsoft
VC++	8.0	IDE	Running Snowglobe	Microsoft
Win SDK	7.0	MS Windows SDK	Snowglobe build	Microsoft
DirectX SDK	9.0	APIs for handling multimedia	Snowglobe build	Microsoft

We incorporated Microsoft Visual Studio 2005 IDE to develop our system, and the primary language used was Visual C++. We adopted Microsoft Foundation Class (MFC) library and an asynchronous socket programming scheme to create a socket-based secure

communication channel. CMake is the main component for building the open source version of the Second Life viewer. Linden Lab, giving the ability to run the Second Life client program into different platforms, provides some intermediate code rather than direct C++ code. With the help of the CMake software tool, we were able to convert the intermediate code. For building the code locally, we also needed to setup operating system SDK. Here, we implemented our system in a Windows machine, and so we installed the Windows SDK and DirectX SDK for the purpose of building the client viewer. With this approach to building the project, we have the flexibility to analyze the communication protocols of Second Life, so that we can attach our add-on on top of this already-existing system. In order to implement voice-based interaction, we used Microsoft Speech SDK (SAPI version 5.1) [33]. Finally, for gesture recognition we used OpenCV, the popular computer vision library tool, originally developed and released by Intel Corporation.

4.2 Platform and Deployment

The prototype is implemented using the latest version of Intel Core i7 PC (860 @ 2.80GHz and 2.80 GHz, 8GB RAM) with the NVIDIA Quadro FX 580 graphics card having 512MB VRAM. The operating platform is 64bit Microsoft Windows 7 Professional Edition. The overall system was first deployed as a Visual C++ 2005 based application, and then converted to an executable setup standalone application which could be operated without the cross-platform build. The later version, unlike the previous one, does not need to setup DirectX SDK, windows SDK, CMake and other necessary applications to build the application in locally. Although we used a high configuration machine, the actual hardware and software requirements depend on the virtual environment being used. For example, the minimum hardware and software requirements for Second Life client installation are 512MB memory, 64MB VRAM, 800 MHz speed processor, and 50 MB hard disk space. To deploy the add-on, user needs to install it on top of an existing virtual environment.

4.3 Realization of Software Architecture

In this section we present some scenario architecture that we implemented by following the architecture described in the previous chapter. At the beginning, in section 4.3.1, we

present our system as applied in an avatar-based interpersonal communication system; later in section 4.3.2, we present the same architecture as used to control a smart home.

4.3.1 Avatar based Haptic Interaction

Our system can be used for an interpersonal communication system for transmitting remote hug, tickle, and touch feelings. In this scenario, two users are each wearing a haptic jacket, and are receiving haptic feelings through their virtual Second Life avatar. Figure 4.1 demonstrates the scenario where the real users are wearing the haptic jacket and they are interacting to each other and receiving remote haptic feelings. This figure also demonstrating how our prototype communicating different modules to give the remote haptic feelings. The interaction controller act the primary roles, it control the whole process to collect the haptic and animation command from the Second Life communication channel and also send to the other user haptic jacket. The interaction controller was developed as a service, which listens to a Communication Serial Port (COM). A Bluetooth device was connected with the PC's USB port, which was virtually configured with the COM port so that the Bluetooth device can send signals to the haptic jacket. For the haptic signal transmission, Bluetooth was configured at the PC COM port of the respective computers that interfaces with the hardware controller of the jacket.

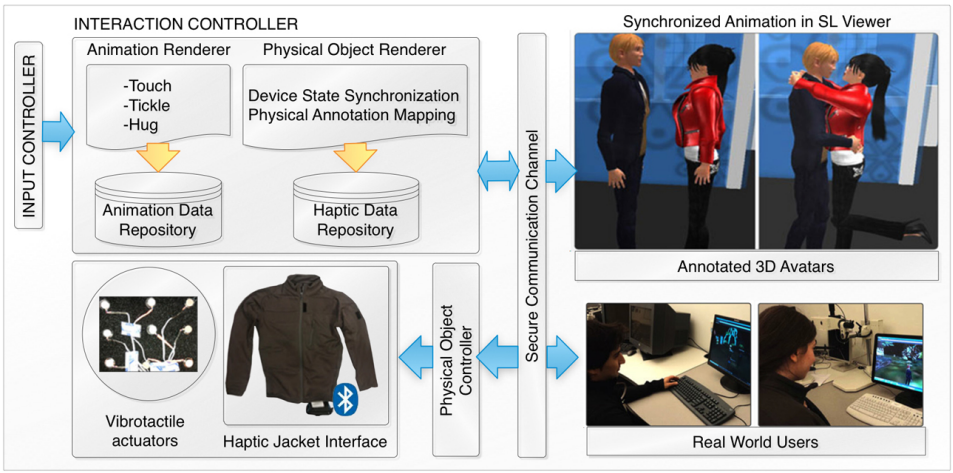


Figure 4.1: A basic communication block diagram depicting various components of the Second Life inter personal haptic communication system.

4.3.2 Object based Smart Home Interaction

Our system can be used in virtual object-based home automation systems, where multiple devices are seamlessly connected through an event-driven architecture. In order to implement this scenario a virtual home is created using Second Life which can mimic the layout of a physical home. The components of the system are depicted in Figure 4.2. In the physical home environment, different devices and sensors are connected in order to ensure a safe and automated home. Any event that occurs in the physical space of the smart home is then synchronized with the virtual environment. More importantly, the virtual home interface provides the option to control the physical smart devices. By using the Second Life virtual interface, the home owner has a better view, and is better able to monitor or control the home appliances.

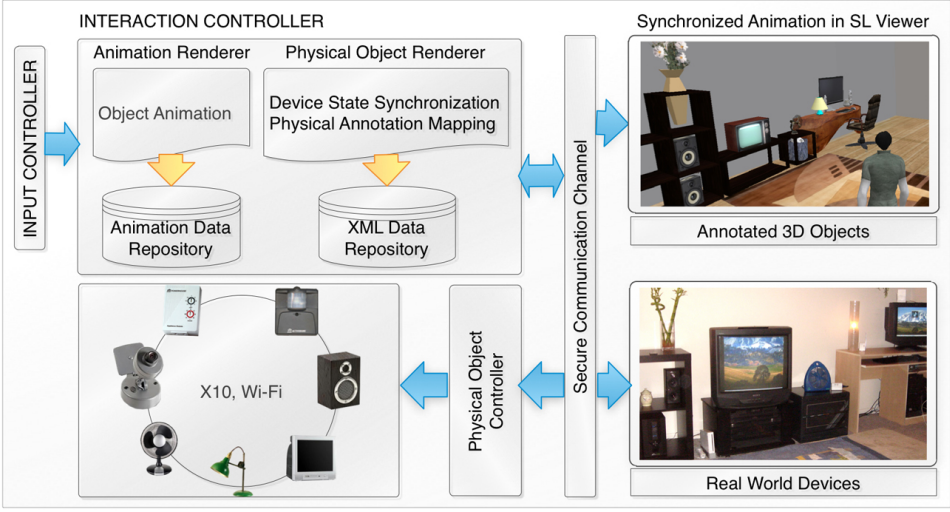


Figure 4.2: An overview of the Second Life home automation architecture. Each physical device has a virtual 3D representation in the Second Life where user can control the physical devices through the virtual objects as well as the physical devices can change the animation by using the home automation services.

The home automation web services and the point-to-point physical connection between services to the wireless and/or wired devices receive messages from the interaction controller. This controller interacts with the other parts of the home automation system by using Simple Object Access Protocol (*SOAP*). The interaction controller receives and/or transmits messages to/from the Second Life communication channel and captures the events that are generated from the message transmission module. When, through the avatar, the user issues events in the 3D environment (such as a click event to open a door)

the message transmission module captures those events and transfers the event messages to the nearby interaction event handler for processing. The interaction controller module determines the particular event by capturing the command from the communication channel. After processing, the controller then sends the packet to a secure communication channel to communicate with the actual home automation services. In the meantime, the synchronization module sends the animation sequences to the Animation Parcel Manager in order to generate an animation sequence for the objects in Second Life.

4.4 Example Interfaces

The example interfaces are organized into two different ways. At the beginning we will present the screen interfaces of haptic and smart home interaction. At the end, we will be showing some essential 2D interfaces, which are mostly related to the interactions with the 3D avatar/object.

4.4.1 Interaction Interfaces

Figure 4.3 depicting Second Life inter personal haptic communication system user interface where two interacting user wearing two haptic jackets and sending hug command.

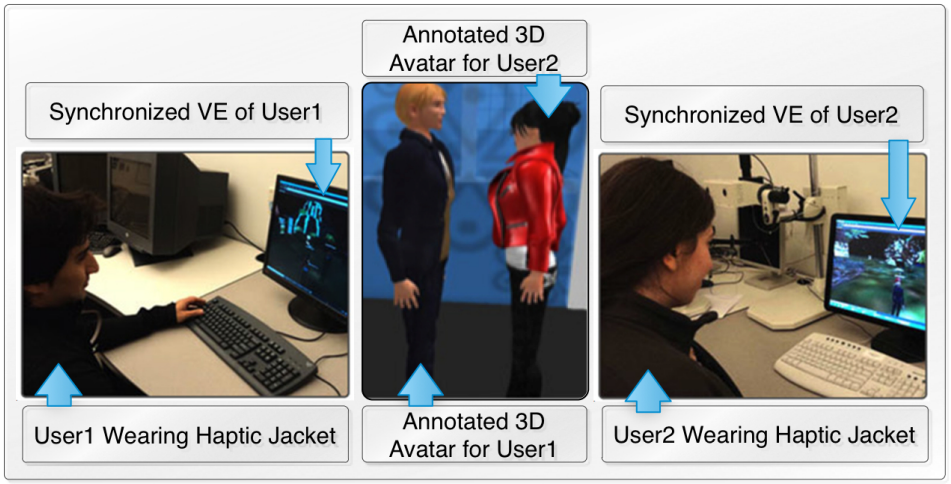


Figure 4.3: Second Life inter personal haptic communication system depicting avatar based hug communication and real world settings.

Figure 4.4 demonstrating the interfaces of Second Life object based home control

where the real world devices can be controlled by using predefined rules that are written in the annotated script in the 3D polygon surfaces. The X10 devices are controlled by the X10 PC interfacing module which is connected with the PC USB and can send or receive signal to or from the X10 modules. The physical objects like fan, television, music system are connected with the X10 modules.

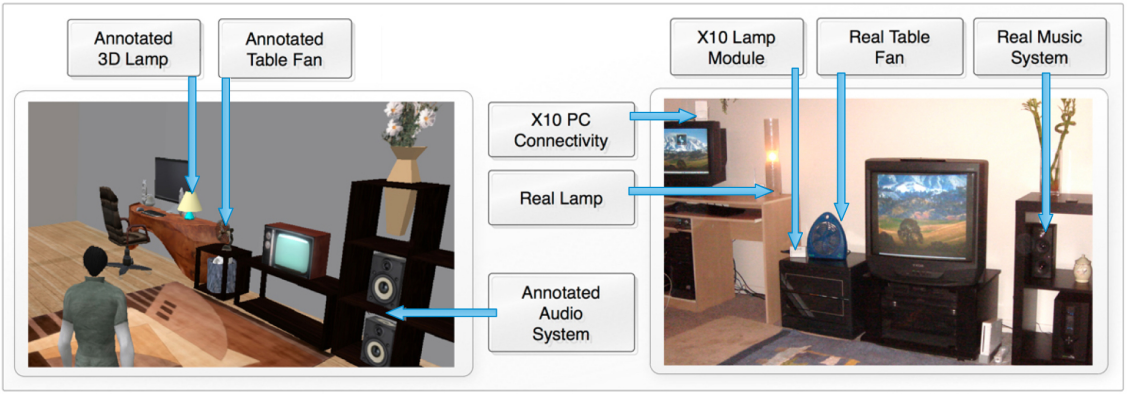


Figure 4.4: Second Life object based home control depicting a virtual world settings mimicking to a real world home.

4.4.2 2D Interfaces

Permission Window

The graphical permission window is responsible for getting the user's choice when any other user wants to interact with his/her avatar. When a user wants to interact (hug, tickle, and touch) with any other avatar, then for security reasons a small window opens to the second users monitor/screen informing them that another user wants to interact. The user has the option to accept, decline or block the user through this window. If the user accepts the message, then the particular interaction that requested by the first user will be initiated. However, if the user blocks it, then the user name will be added to a 'block list', although individuals have the option of later removing a user name from their blocked list, and can then receive haptic interaction signals from them again. If the user accept the message then the particular interaction that requested by the first user will be initiated. However, if the user block the user then the user name will be added to a block list.

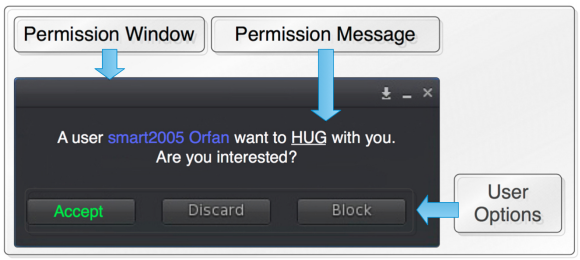


Figure 4.5: Permission Window that used to get the user's choice when any other user wants to interact with his/her avatar.

Nearby Chat Bar

This is a text box for chatting with other avatars. All the text that is typed in this bar will be sent to the client who is within a 10-meter distance (this is the area of interest distance for Second Life). We use this nearby chat bar for exchanging text-based commands through keyboards. Although users have the ability to click for touch or tickle interaction, this text bar is especially important for hug-type interactions. If the user wants to initiate a HUG, then both the users have to come within a certain distance of each other. The distance must be within the area of interest, which is 10 meters for Second Life.

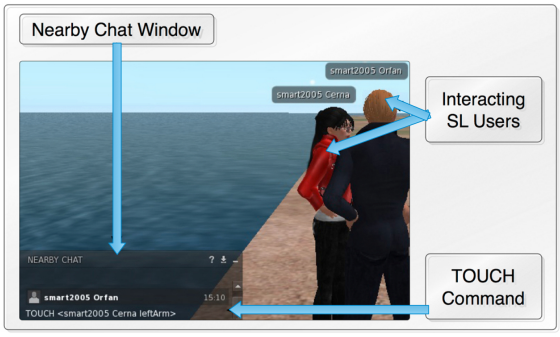


Figure 4.6: Nearby Chat Bar.

4.5 Development of Different Modules

Here, we present the details of the implementation issues of different modules of our proposed system.

4.5.1 GUI Control Module

The GUI controller enables the usage of keyboard-, mouse-, speech- and gesture-based inputs from the user. For text based commands, we used Second Life's nearby chat bar, through which users can send or receive chat messages to or from other users. In order to incorporate mouse-based interaction, we used the Second Life scripting facility, whereby users can annotate custom script to a 3D polygon surface. Based on the content of the script, it generates events, and our add-on captures the message signature from the communication channel. We incorporated the existing speech-based interaction methodology in the Second Life virtual environment [14] and Microsoft SAPI SDK for implementing the speech facility. As user name recognition was not attempted in our system, we recognize only some predefined name tags. Finally, for gesture recognition we incorporated our existing motion path-based gesture interaction system [35]. In order to recognize/for the purpose of recognizing the motion path, we used the popular API called OpenCV. In the motion path-based recognition we capture one drawing symbol at a time and store it into a queue; when any sequence already matched with the predefined sequences occurs, then that particular command is initiated.

4.5.2 Message Transmission Module

The Message Transmission Module (MT) captures all the messages that are generated from the Second Life Nearby Interaction Event Handler in an encrypted XML [48] format. The primary responsibilities of the module are to decrypt those received messages and transmit them further down the communication channel towards the Access Control Module. All the messages that are exchanged between server and client are in a predefined format. Based on our open source viewer analysis, we found out that Second Life uses its own messaging protocol. The Message Transmission Module captures those messages and parses them to identify the event types.

4.5.3 Access Control Module

In our system user have the facility to incorporate user profile specific access control mechanism in order to provide the participating users the means of authenticating and personalizing their interactions. The Access Control Module looks up the user dependent access control scheme and produces appropriate permission dialogues in Second Life viewer. The permission manager issues this dialog by using SL script and receives

appropriate permission parameters. Figure 4.7 shows the code snippet that is used to control user dependent animation and the vibrotactile motors in the haptic jacket. As shown, before commencing avatar or haptic rendering functions, we call *llRequestPermissions(key AvatarID, integer perm)* function. The function takes two parameters; the first parameter is the user's Avatar ID who requested an event. The second parameter *PERMISSION_TRIGGER _ANIMATION* is a permission type for that event.

```
state_entry()
{
    //Request animation permission to Second Life
    llRequestPermissions(llGetOwner(), PERMISSION_TRIGGER_ANIMATION);
    llSetTimerEvent(1.0);

    //Specify communication channel
    llListen(&iChannel,"",llGetOwner(),"");
}

touch_start(integer WHICH)
{
    //Obtain the calculated body part from the GUI interaction
    llGetAnnotationSelection(iChannel,&vBody_trg,
                                (string)llDetectedKey(iChannel));

    //User specific haptic feedback and animation
    aAnimations = SL_GetUserAnim(SL_GetUser(this),vBody_trg);
    hFeed = SL_GetUserFeed(SL_GetUser(this),iChannel,vBody_trg);

    //Render animation and haptic feedback
    llStartAnimation(llList2String(aAnimations,WHICH));
    SLStartFeed(llList2String(hFeed,WHICH));

    //Animation calibration and user interaction throttling
    if(WHICH++ >= TOTAL)
    {
        WHICH = INITIAL_VALUE;
    }
```

Figure 4.7: A code snippet depicting portion of the Linden Script that allows customized control of the user interaction and permission.

4.5.4 Animation Parcel Manager

An Animation Parcel Manager (*APM*) is a container that contains the BVH file and the associated resources for playing the animation. In the implementation of the APM, we used Microsoft Foundation Classes (MFC) where the container is a Search Results serializable vector list. When any resource command was received from the message transmission module, it then would search for/find the associated resources in the repository and add those resources (like BVH, XML file) to the vector list. When all the re-

sources are loaded successfully to the list, then a unique ID is attached to the container, and sterilization is performed; this sterilized text is then passed to the communication channel.

4.5.5 Device State Synchronization Module

We implemented the synchronization algorithm that we presented in the previous chapter (algorithm 2) in Microsoft Visual C++ version 6.0. At the beginning all the device names and their associated IDs are stored into a XML file. We can get a device ID via the function $getDeviceIdByName(devicename)$. An animation has three main states, which are called: PLAYING, STOPPED, and CRASHED. When an animation is running in the VE, then it is in PLAYING state. However, the PLAYING state has different positions that indicate the exact stage of the animation it is currently playing. For example zero (0) position means that the animation is just being started; and when the position is equal to the total length of the animation, this indicates that the animation is about to come to an end. We implemented two methods for getting and setting this animation state, called $getAnimationStatus()$ and $resetAnimation()$. Similarly for X10 devices we implemented three functions: $setState()$, $getState()$, and $getLevel()$, for setting a state to the X10 device, getting the state of a device and getting the current level of a sensor.

4.5.6 Physical Annotation Mapping

In the prototype system all the events are controlled by predefined names such as: hug, tickle, light, fan, and music player. But this is not the actual device ID. In the system we also provide an extra level of mapping between these names and the actual device ID. Figure 4.8 demonstrates some sample mapping entries. This XML also has the duty of giving some extra facilitiesfor example, if users want to auto-lock a door (in the real world, it is very common that when a person enters into a secure place through his or her RFID, after a few seconds the door will be locked automatically), the necessary configuration data is stored in this XML file.

4.5.7 Secure Communication Channel

We implemented the secure communication channel by using MFC socket class. The communication channel has two parts, both parts act like a peer. When any information receives one part then it sends to the other part. In order to ensure security before sending

```
<!-- VLC section-->
<add key ="VLC_SnapshotHTTPCommand"
                value ="http://localhost:8080/requests/status.xml"/>
<add key ="VLC_Path" value ="C:\Program Files\VideoLAN\VLC\vlc.exe"/>
<add key ="VLC_SnapshotFile" value ="\CameraImages\newPicture.jpg"/>
<!-- Logger section-->
<add key ="SnapshotLogFolder" value ="\CameraImages\Log\"/>
<add key ="SnapshotLogFile" value ="\CameraImages\Log\log.txt"/>

<!-- Door section-->
<add key ="DoorUnlockDurationMilliseconds" value ="15000"/>
<!-- X10 section-->
<add key ="EntranceMotionSensorAddress" value ="A1"/>
<add key ="EntranceLightSensorAddress" value ="A2"/>
<add key ="EntranceDoorButtonAddress" value ="A7"/>
<add key ="EntranceLampAddress" value ="A10"/>
<add key ="EntranceDoorLockAddress" value ="A9"/>
<add key ="DoorBellNotificationAddress" value ="A11"/>

<!-- Mapping Appliances -->
<add key ="Light1" value ="A10"/>
<add key ="Door1" value ="A9"/>
<add key ="Fan1" value ="A1"/>
```

Figure 4.8: Physical annotation mapping that can give a user extra layer of mapping facility.

any messages to the channel it automatically encrypts the data that passes through it and decrypts the data in the recipient section automatically.

4.5.8 Haptic Jacket Controller

The Haptic Renderer (*HR*) operates the haptic jacket and notifies the Animation Parcel Manager for synchronized animation feedback. In order to control the jacket motors, it parses an XML file containing haptic patterns and sends a message to the micro-controller unit of the jacket accordingly. Portion of the xml file is shown in Figure 4.9. In our implementation the actuator motors have a total of 16 intensity levels from 0 to 15. Where, 0 means no vibration and 15 indicates the maximum vibration level. To repeat the vibration patterns we set the value for the *numberOfRepetation* attribute.

4.5.9 Home Automation Services

For the home automation service we used ActiveHomeScript Library [52] and Web Services Dynamic Discovery (*WS-Discovery*) which is a multi-cast discovery protocol defined with the purpose of allowing dynamic discovery and advertisement of target services willing to find a specific service. It can query the network using multi-cast search messages,

```
<?xml version="1.0"encoding="utf-8"?>
<InteractionRules>
   <hug userType="Friends"hapticFeedback="Yes"name="hug1">
     <animationModules>
       <UUID>6b61c8e8-4747-0d75-12d7-e49ff207a4ca</UUID>
       <animationBVH>hug.bvh</animationBVH>
       <animationPriority>MEDIUM</animationPriority>
       <animationLooped>YES</animationLooped>
       <animationSpeed>30</animationSpeed>
       <animationDuration>LOW</animationDuration>
       <animationScaleTo>0.75</animationScaleTo>
     </animationModules>
     <tactileModules>
       <module name="leftChest">
          <highestIntensity>15</highestIntensity>
          <lowestIntensity>0</lowestIntensity>
          <vibrationType>GRADUAL_INCREASE</vibrationType>
          <interactionTime>500</interactionTime>
          <numberOfRepetation>3</numberOfRepetation>
       </module>
       <module name="rightChest">...</module>
     </tactileModules>
   </hug>
   <hug userType="Family"hapticFeedback="Yes" name="hug2">...</hug>
   <touch>...</touch>
   <tickle>...</tickle>
```

Figure 4.9: An overview of the target user group specific interaction rules stored in an xml file.

and services satisfying the query should reply. We also incorporated web services eventing (*WS- Eventing*), which is a web service protocol that describes how a client (subscriber) can register for some events (subscriptions) of a web service (event source). Thus, any clients can be notified about changes in the service, without requiring a standard polling mechanism. At the outset, when any device wants to connect with the web service, it needs to send multi-cast messages to search target services by type or within a certain scope. Then, a uni-cast response is sent to the sending client when the target service matches a Probe message. In this way, after the connection is established successfully, the device needs to subscribe to an event in order to send messages to/receive messages from the server. In our prototype we used the services that are listed in Table 4.2.

Table 4.2: List of services that were running in the prototype system.

Service Name	**Service Description**
Entrance Manager	Notifier Visitor, Unidentified Person arrival. Open door, close door event
Notification Manager	Notify any illegal event occur inside the home
VLC Controller	Observe activity of users inside the home using VLC player
RFID Reader	Receive RFID [51] tag data and send to the server
X10 Controller	Send or receive x10 command or query

4.6 Summary

In this chapter we presented the implementation process of the prototype system. Additionally, we presented the implementation details of the software architecture, platform and deployment issues, and some sample interfaces.

Chapter 5

Evalution and Results

To evaluate the user's quality of experience with the prototype and to justify the suitability of our proposed approach we have performed several quantitative and qualitative measurement studies. The quantitative analysis of the prototype is performed by evaluating the processing time and the response time of different modules of the system. On the other hand, the qualitative analysis is performed by studying different usability aspects of the proposed system. Section 5.1 and 5.2 presents the quantitative and qualitative studies respectively.

5.1 Quantitative Measurements

In this section we present the processing time of different modules and the interaction response time calculation in section 5.1.1 and 5.1.2 respectively. As the system performance changes when multiple users use the system concurrently. We discuss this multi-user access performance and usages of the system, in section 5.1.3. Finally, in section 5.1.4, we illustrate a detail analysis of the impact of different interaction modalities.

5.1.1 Processing Time of Different Modules

In order to ensure that the implemented interacting modules of the system performs on par with the interfacing modules of Second Life, we measured their performances with a set of haptic and animation data. The data size for the haptic and animation rendering in each step of the experiments were kept the same for all the components. In order to measure the performance metrics we embedded performance thread hooks in the components and recorded the responses of those. For each pair of haptic and animation

rendering the experiment setup were repeated for twenty eight times. The result of our observations are depicted in Figure 5.1. This figure illustrates the performance of the different interacting modules of our system.

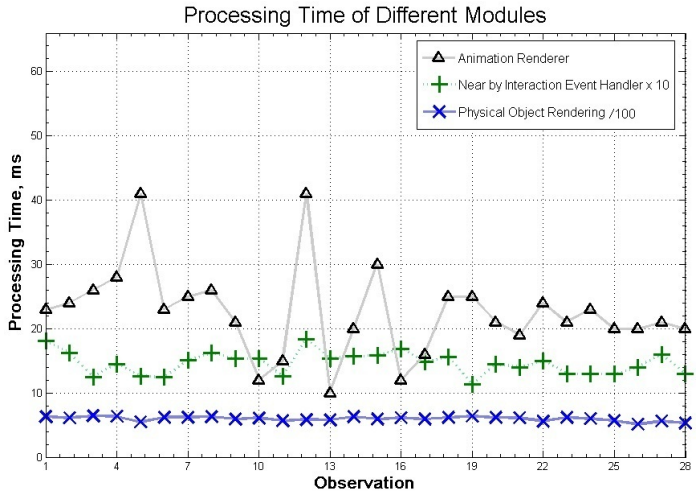


Figure 5.1: Processing time of different interacting modules of the system.

The overview of the processing time of different interacting modules are listed in Table 5.1. In this table low standard deviation indicates that the data points tend to be very close to the mean, whereas high standard deviation indicates that the data points are spread out over a large range of values.

Table 5.1: Overview of the processing time of different interacting modules.

Interacting modules	**Mean (ms)**	**Std. Dev.**
Physical Object Renderer	596.8	33.22
Near by Interaction Event Handler	1.5	0.2
Animation Renderer	22.6	7.1

As shown in the table the nearby interaction event handler required average 1.5ms to compute the interaction and report that to the animation renderer. The core components e.g. message transmission, permission manager and animation parcel manager processed the message data on an average 22.6ms. Finally the physical object renderer (Bluetooth transmission or X10) render complete the execution on an average 596.8ms. This physical object renderer took more time to process the request compared to the other modules because physical renderer render their operations locally and dependent on their hardware processing time of the Bluetooth or X10 modules.

5.1.2 Interaction Response Time

We calculate the transmission time from the sender machine to the receiver device (haptic jacket or the *X10*) by using Equation 5.1. Where user's average interaction time to interact with the 3D virtual environment client (e.g. Second Life viewer) is I unit, average data transmission rate via the server is Π, n is the message size and the time for sending data from the receiver machine to the receiver device (e.g. jacket actuators) is one β_1 unit.

$$R = I + \frac{n}{\Pi} + \beta_1 \tag{5.1}$$

After generating an interaction event the system approximately requires $\mathbf{R} = (3775 + 270 + 344)ms$ to complete the transmission. Here, in our experiments the average of the interaction time is $3775ms$, network overhead is $270ms$ and β_1 is $344ms$. The acknowledgment (haptic or *X10*) from the receiver machine to the sender device i.e. haptic jacket or the home service is represented by equation 5.2. Here, $\frac{n}{\Omega}$ is the average time for transmitting n byte feedback message from the receiver machine to the sender machine. We assumed that the transmitted message and its acknowledgment were of the same length.

$$\begin{aligned} S &= I + \frac{n}{\Pi} + \beta_2 + \frac{n}{\Omega} \\ &= R + \frac{n}{\Omega} + (\beta_2 - \beta_1) \\ &= R + \frac{n}{\Omega}, \qquad (\beta_2 - \beta_1) \simeq 0 \end{aligned} \tag{5.2}$$

On average, the time S is higher than R by $\frac{n}{\Omega}$ unit, which is the network transmission delay. In order to ensure that the difference between S and R does not affect the interaction, the object rendering and animation rendering are synchronized locally in respective users' machines.

The interaction response time depends not only on the network speed but also depends on the haptic data size, the receiver device types. We calculate the interaction response time by varying this factors. Figure 5.2 depicts the interaction response times required to render different haptic data. From the result we see that hug interaction needs more time than other interactions as for the hug type rendering the system is required to process more data than the others. Table 5.2 illustrates the overview of this response time.

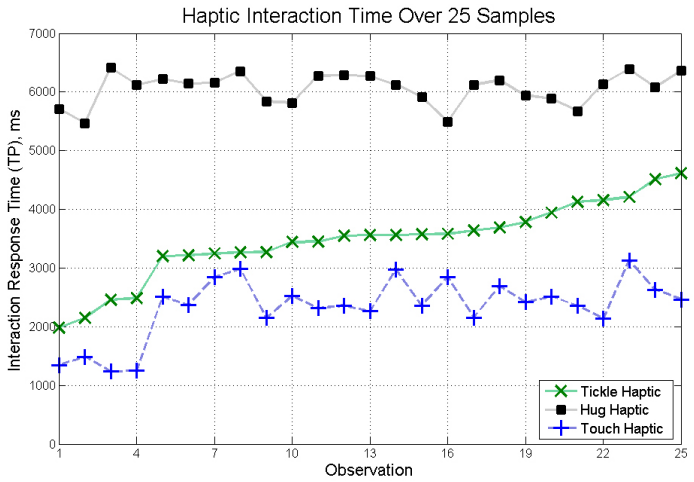


Figure 5.2: Haptic rendering time over twenty five samples.

Table 5.2: Overview of the response time of different haptic interactions.

Haptic type	**Mean (second)**	**Std. Dev.**
Tickle	3.47	0.66
Hug	6.06	0.27
Touch	2.33	0.52

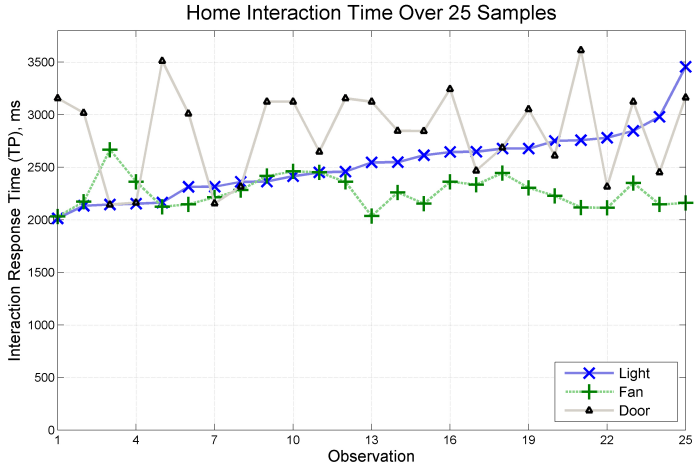


Figure 5.3: Interaction time of different smart home devices over twenty five samples.

Finally the figure 5.3 depicts the interaction response times required to render different objects that are connected with X10 modules. Controlling a door require more time (average $2844ms$) that fan or light control, because for annotating a door we attached extra rules (the door will be closed automatically and store which user are entering the door). These extra rules need some processing time. Table 5.3 shows the average interaction response time and the standard deviation of different smart home devices.

The Second Life message transmission architecture plays a role to introduce delay in the synchronized interaction and animation rendering thereby increasing the interaction processing time. We present two main factors that we observed during our experiments. We found out that Second Life nearby message transmission component introduces delay in its message processing when the server receives too many requests from the surrounding of the avatar. To measure the difference in the processing times we designed experiment sessions on five empirically selected time intervals during a day. We continued to sample the interaction time responses in three successive weeks by running the same set of experiments. We show the recorded data in the following Figure 5.4. As seen in the figure, the interaction time responses reached its pick during the weekends.

Table 5.3: Overview of the response time of different smart home devices.

Smart home device	**Mean (second)**	**Std. Dev.**
Light	2.52	0.32
Fan	2.27	0.15
Door	2.84	0.42

In Second Life the users can navigate to different map locations in the virtual world. A convenient method of specifying locations and teleporting to that location is achieved by using the slurls [42], which are hyperlinks that allow users login directly to that site or teleport to it if they are already inside Second Life. The basic format for a SLurl is: $[slurlDomain]/ < region > / < x > / < y > / < z >$, Where: $< region >$ is the name of the destination region, $< x >$ and $< y >$ are the east/west and north/south coordinates of the destination; $< z >$ is the vertical (height) coordinate. We noticed that different slurls have different 3D object density (called prims, or primitives) and they are spanned in varying size. A single shape is one prim and can be linked with other objects to make up many prims. A chair, for example, might take up 11 prims. When the number of prims increases the interaction complexity with the present virtual avatars in that area increases. These metrics therefore, influences the interaction response time in

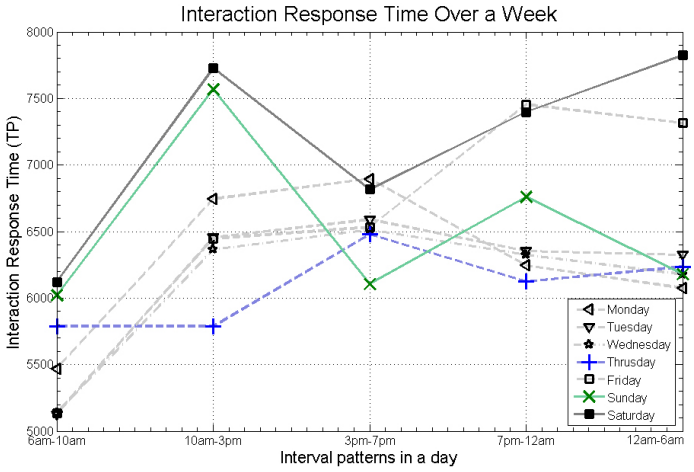


Figure 5.4: Average of the interaction response times that were sampled on particular time intervals. The data were gathered during three weeks experiment sessions and averaged. From our analysis, we observed that based on the server load the user might experience delay in their interactions.

our experiments as shown in table 5.4. The area of the slurl effectively creates different density of the avatars with particular prims in their surrounding virtual locations. In order to measure the effect of different density levels we teleported the avatars in locations with very low to very high density and determined their impact on the interaction response time. Our findings are depicted in figure 5.5.

We equipped the interaction controller to sense these network parameters. By sensing the density of the avatars, prims, nearby interaction message parsing frequency and day time metrics the interaction controller empirically calculates a threshold that can distinguish disruptive effects of network lag in the inter-personal communication. However, as our system extensively used the communication platform of the Second Life controller we tackled the lag at the network communication level by deciding to inform the participating users about the delay. The interaction controller incorporates the calculated lag threshold and provides color-coded decorators at the Second Life viewer's Heads Up Display (HUD). In this regard we adopted the decorator scheme proposed in [41]. We developed a bar in the HUD that displays green when the prims and avatars interaction messages do not create congestion. Similarly a red bar is displayed affecting the delay in the communication. Later, in our usability study we noticed that when the user was informed about possible interaction delays by using the decorators s/he accepted the lag with ease and reacted more intuitively.

Table 5.4: Interaction response time based on Prim size on various Second Life map locations.

Location (Slurl) [42]	Area (m2)	Prims on parcel (Object Density)	Interaction Response Time (TP), ms
OakGrove/128/128/11	528	600	5120
SaintLucia/128/128/22	1792	269	5469
Amberville/128/128/2	5472	1391	6089
Boreal/128/128/122	7168	1495	6201
KissenaPark/128/128/2	8192	1426	6213
Wichi/128/128/2	9408	2018	6213
MooseBeach/50/57/20	19008	1382	6428
NewYorkNYC/128/128/2	21936	4341	6901
ZenDestani/128/128/18	28672	2556	6310
SolaceBeach/128/128/2	38832	4911	6052
LondonUK/128/128/22	47760	5460	7168

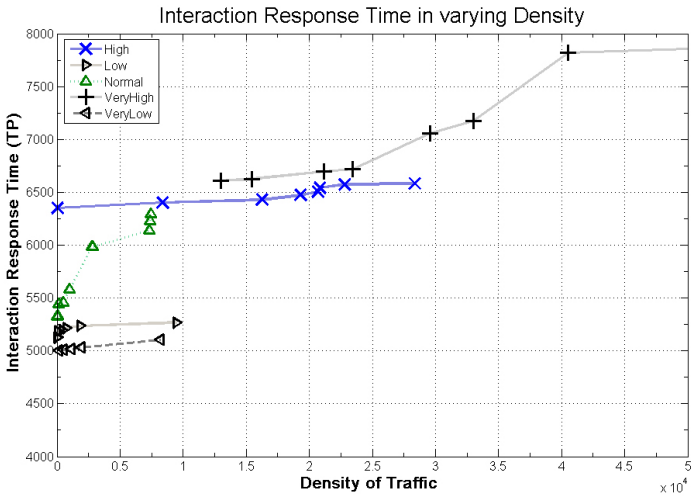


Figure 5.5: Interaction response time in varying density of traffic in the Second Life map location.

5.1.3 Multi-user Interaction Response Time

The real object and animation rendering based communication is not specific to a pair of users. Rather, multiple users from different groups can interact with each other at the same time. This essentially extends the inter-personal interaction and provides option to leverage the framework in a group specific interaction scenario. The developed Interaction Listener supports interaction requests from multiple users. For example, when a user A, receives interaction request from user B and C, it creates of queue of request for user A on a first come first serve (FCFS) basis. In such cases the Interaction Response Time can be calculated by using the Little's Formula, which is a classical conservation equation in queuing theory.

$$E(T) = \frac{E(n)}{\lambda} \tag{5.3}$$

Here, $E(T)$ is the average delay for a user request, i.e. average throughput time, $E(n)$ is the average number of requests to the interaction listener. λ is the arrival rate of the requests. However,

$$E(n) = \frac{\rho}{1-\rho} \; and \; \rho = \frac{\lambda}{\mu} \tag{5.4}$$

Where, ρ is the fraction of time the interaction listener requires to process the requests. Hence, combining Equation 5.3 and 5.4 we get,

$$E(T) = \frac{\rho}{\lambda(1-\rho)} = \frac{\frac{\lambda}{\mu}}{\lambda(1-\frac{\lambda}{\mu})} = \frac{1}{\mu-\lambda} \tag{5.5}$$

Here, μ is request processing rate from the queue, and Equation 5.5 is the queuing delay. In this equation λ is the arrival rate of an interaction request by a user and μ is the average service rate. From Figure 5.1 we measured that the average service time is approximately $6.5 seconds$. Hence, average service rate $\mu = \frac{1}{6.5} = 0.1538$ per *second*. For example, in case after every $60 seconds$ an interaction requests is triggered to the Interaction Controller then $\lambda = \frac{1}{60} = 0.0167$ request per *second*. Therefore, from the Little's Formula (Equation 5.5), the total waiting time including the service time $= \frac{1}{0.1538-0.0167} = 7.29 seconds$ (approx.).

5.1.4 Analysis of Different Interaction Modalities

A comparison of different interaction modalities used and their suitability for each haptic interaction are given in Table 5.5. The two other parameters are average time needed to produce the command and average accuracy, which are also listed? However, not all haptic input commands were convenient to use for each interaction modalities.

Table 5.5: A comparison of different interaction modality

Modality	Average Time	Accuracy	Suitability
Keyboard	5110 ms	85%	hug, touch, tickle
Mouse	2075 ms	99.6%	touch, tickle, hug
Speech	3790 ms	55%	hug, tickle, touch
Gesture	4125 ms	78.1%	hug, touch, tickle

For the keyboard (text) interaction modality we found that writing body parts names take time and often spelling mistakes impaired the accuracy of the command. Touch and tickle input commands were very easy to issue using the mouse-based modality. However, while issuing hug input command using the mouse, it became difficult to assign the command to a particular user, hence nearest user was selected automatically from the user group lists. Similar problem occurred while using speech and gesture based interaction modalities as it became cumbersome to recognize the user names using either of those two approaches. From the table we see that the percentage of accuracy is highest for mouse-based interaction modality, which is 99.6%. This and its flexibility for usage in pointing and interacting with annotated body parts made it the ideal medium for haptic input command delivery in our system.

5.2 Qualitative Measurements

We have incorporated the usability evaluation guidelines [10] [43] to qualitatively measure our proposed system and designed our tests accordingly with the sensory analysis [24] of the system involving both the user and the targeted sensory communication modules. Before performing the usability test we designed a test plan where we defined our evaluation objectives, developed questions for the participants, identified the measurement criterion and decided upon the target users of the system. The test took place at our university laboratory with sixteen (16) participants comprising of different age groups.

Five (5) of the participants are in age group 13-18, eight (8) of them are in age group 18-36 and the rest three (3) are in age group 36+. Furthermore, the users were divided into two groups namely Group A and Group B.

Table 5.6: Usability test questions to the user for haptic interaction.

#	**Question**
Q1	Perceived system response was acceptable
Q2	The device feedbacks are realistic and/or acceptable
Q3	Consider using the system in 3D Virtual Environment like Second Life
Q4	Perceived delay between response and avatar rendering was tolerable
Q5	Easy to get familiar with

For the traditional experiments two users were chosen. In order to ensure that each communicating participant can converse with different age groups their selection was made randomly. Moreover to ensure the distributed communication behaviour the physical location of the users were separated. In a user's test machine the enhanced Second Life viewer with the add-on was installed to provide animation and GUI based interactions. At a time, the selected volunteers were told to put on the haptic jackets and requested to use the prototype system by participating in certain haptic interaction based tasks. Their activity was monitored throughout the experiment and recorded for analysis. Afterwards, they test our home control scenario. In this scenario we annotated two fans, three lights, and one door object in the Second Life. Those objects are installed into two separate rooms. Finally, based on their test experiences the users filled out a questionnaire where they were requested to provide ratings of their likeliness, familiarity, ease of usage etc of the system.

Table 5.7: User satisfaction on the overall evaluation in Likert scale.

	Mean	**Std. Dev.**	**Mean Percentage**
Acceptability	4	0.7303	80%
Device Feedback	2.6875	1.0782	53.75%
Likeliness	3.8750	0.8062	77.5%
Delay	3.875	1.3102	77.5%
Ease of use	4.25	1	85%

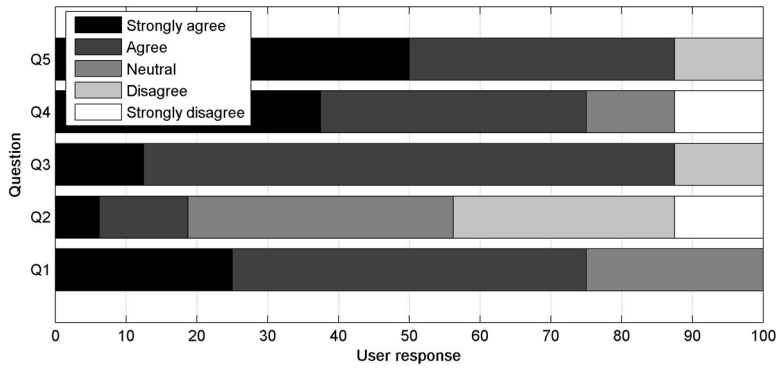
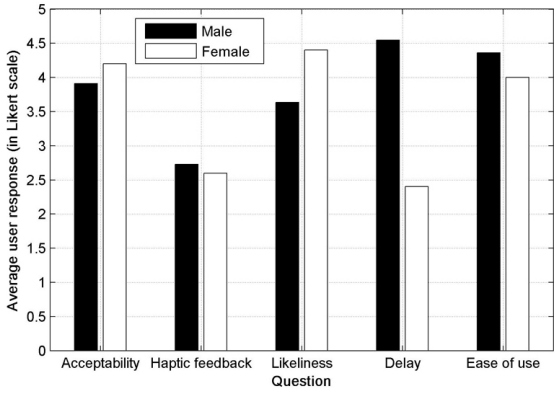


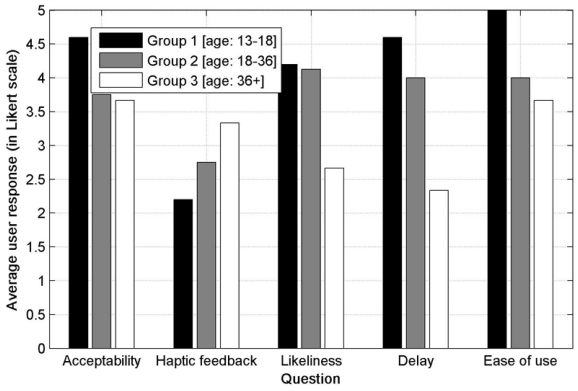
Figure 5.6: User response for the usability test questions that are listed in Table 5.6.

The user responses are shown in Likert Scale [31] in Figure 5.6. The ratings of the questionnaire were in the range of 1-5 (the higher the rating, the greater is the satisfaction). The average of the responses of the users were calculated in percentage form and measured after the usability tests. Figure 5.6 shows the user's responses for each given assertions. It is worth mentioning that more than 80% of the users would like to use the enhanced system if they were available in Second Life. Overall the users were also satisfied with the synchronized animation and real world responses and 75% of users consented to that. Table 5.7 summarizes the overall performance score of the users. The higher mean values of System response, device feedbacks, and Easy to familiar represent a very satisfactory user response, while the moderate mean values of System in Second Life and Perceived delay show relatively good user satisfaction.

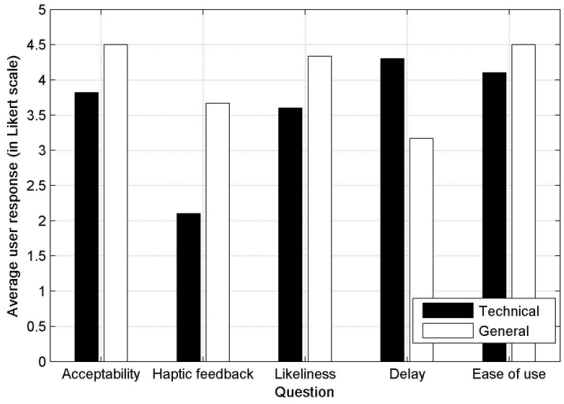
In our study we also attempted to evaluate the acceptability of the system by the users from different genders, age groups and technical backgrounds. The result of these studies is depicted in Figure 5.7. We infer from the Figure 5.7 (a) that the female users gave more positive feedback in acceptability and likeness than the male users. However, the male users confirmed that it was easier for them to use the system after a couple of dry runs. Moreover, all the users favoured that a refined device rendering is needed to make the interaction experience natural and realistic. In case of different age groups we divided the users into three age groups, namely $group-1$: ages $13-18$, $group-2$: ages $18-36$, and $group-3$: ages 36+ and recorder their responses in Figure 5.7 (b). In retrospect, compared to the older group of users, the users from the younger group seemed to be more attracted in using the system and wanted to participate in remote touch, hug, and tickle interactions. Also from Figure 5.7 (c) we received favourable responses and recommendations from users with non-technical background than that of



(a) Gender



(b) Age Group



(c) Technical Background

Figure 5.7: Comparison between the responses of users from different (a) gender (b) age group and (c) technical background.

the technical people. Although non-technical users were less happy with the interaction response time of the system.

5.3 Summary

In this chapter we presented the quantitative and qualitative measurement studies of our prototype system. In the quantitative study we presented interaction response time both for single user and multi-user scenario. We also presented how different interaction modalities affect the system performance. To evaluate the user's quality of experience with the prototype we presented user studies based on different user groups including gender, age and technical knowledge of the system.

Chapter 6

Conclusion and Future Work

In this chapter, we first review the works presented in this thesis. Then, we discuss some topics that we did not treat or that could not be treated in more detail. First in Section 6.1 we present the concluding remarks of our research that we described. After that, in Section 6.2, we present some future directions for inquiry that can help to expand upon this work.

6.1 Conclusion

The virtual environment, by default, has its own attractions to the Internet user. Specially for young people, the technology-mediated communication of human emotions and related feedback through the virtual world is proving to be of great importance. These emotional feedback cycles, that are fundamental to physical and emotional development, can in turn enhance the user's interactive and immersive experiences with virtual social communities. We explored the possibilities of integrating haptic and real world object interactions with Linden Lab's multiuser online 3D virtual world, Second Life. We enhanced the open source Second Life viewer client in order to facilitate communications between the real and virtual world. The system we developed works as an add-on, and is loosely coupled to the Second Life viewer. The haptic and animation data are annotated in the virtual 3D avatar body parts. The 3D avatar and the annotated body parts representing a real user receive inputs when they are interacted with through gesture-, mouse-, speech- or text-based input modalities, and produce emotional feedback such as touch, tickle and hug to the real user through the haptic jacket. The animation and device control data were annotated in the virtual 3D object in Second Life. The 3D object,

representing a real device, received input when interacted with a user; it automatically responded to changes in the state of the physical environment. Finally, we showed some test statistics covering different quantitative and qualitative measurement aspects of our system. Our experiment suggests that the proposed approach not only demonstrates an intuitive type of communication system, but is also appealing and useful to the user.

6.2 Future Work

While we tried to address as many issues as possible related to the design of a prototype for virtual/real communication, there are still some topics that we did not treat or that could be treated in more detail. There are some limitations in our prototype tool. One of the biggest limitations of our system originates in the fact that since it follows the peer-to-peer communication approach, there should be one-to-one mapping between the virtual and real world objects or devices; it is therefore not possible to send one-to-many or many-to-many based commands. Due to this limitation, users have to know which annotated part is representing which device in the physical world; this requires extensive concentration. Another limitation is that although we handled device state synchronization with the animations played in the virtual environment that would not work for all the cases. For example, it is much easier to open a door through the virtual environment, but what happens is that if someone opens the real door through the RFID tag, it will not send any message to the virtual world, so no animation will be played in the virtual world. With our prototype we can control only those devices that have fixed states, like on/off, up/down etc.

In addition, we will conduct further investigations on more realistic haptic interaction that can differentiate between the types of object the avatars are touching. We are also planning to add a functionality by which the system can track a moving object or human activities. A potential example is of parents who want to track the behaviour of their children. However, we believe that our proposed techniques for communication between the virtual and real world will remain as our motivation for further research in this area.

Reference

[1] R. Antonello, S. Fernandes, J. Moreira, P. Cunha, C. Kamienski, and D. Sadok. Traffic analysis and synthetic models of second life. *Multimedia Systems*, 15:33–47, 2009. 10.1007/s00530-008-0125-1.

[2] W. Barfield and C. Rosenberg. Judgments of azimuth and elevation as a function of monoscopic and binocular depth cues using a perspective display. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 37(1):173–181, 1995.

[3] A. Barghout, J. Cha, A. El Saddik, J. Kammerl, and E. Steinbach. Spatial resolution of vibrotactile perception on the human forearm when exploiting funneling illusion. In *Haptic Audio visual Environments and Games, 2009. HAVE 2009. IEEE International Workshop on*, pages 19 –23, 2009.

[4] J. Bishop. Enhancing the understanding of genres of web-based communities: The role of the ecological cognition framework. *International Journal of Web Based Communities.*, 5(1):417, 2009.

[5] Blizzard Inc. Online game store. Technical report, http://us.blizzard.com/store/index.xml, Last accessed, August 2011.

[6] L. Borodulkin, H. Ruser, and H.-R. Trankler. 3d virtual "smart home" user interface. In *IEEE International Symposium on Virtual and Intelligent Measurement Systems VIMS '02*, pages 111 – 115, 2002.

[7] M. N. K. Boulos, L. Hetherington, and S. Wheeler. Second life: an overview of the potential of 3-d virtual worlds in medical and health education. In *Health Info Libr J*, volume 24(4):233-45, 2007.

[8] S. Brave and A. Dahley. intouch: a medium for haptic interpersonal communication. In *CHI '97 extended abstracts on Human factors in computing systems*, pages 363–364, New York, NY, USA, 1997.

[9] J. Cha, M. Eid, A. Barghout, A. Rahman, and A. El Saddik. Hugme: synchronous haptic teleconferencing. In *MM '09: Proceedings of the seventeen ACM international conference on Multimedia*, pages 1135–1136, New York, NY, USA, 2009.

[10] D. Chisnell. Usability testing: Taking the experience into account. *Instrumentation Measurement Magazine, IEEE*, 13(2):13 –15, 2010.

[11] A. Cockburn and B. McKenzie. Evaluating the effectiveness of spatial memory in 2d and 3d physical and virtual environments. In *Proceedings of the SIGCHI conference on Human factors in computing systems: Changing our world, changing ourselves*, pages 203–210, 2002.

[12] M. de Pascale, S. Mulatto, and D. Prattichizzo. Bringing haptics to second life for visually impaired people. *Haptics: Perception, Devices and Scenarios*, pages 896–905, 2008.

[13] C. DiSalvo, F. Gemperle, J. Forlizzi, and E. Montgomery. The hug: an exploration of robotic form for intimate communication. In I. Press, editor, *RO-MAN 2003*, pages 403–408, 2003.

[14] A. El Saddik, A. Rahman, and M. Hossain. Suitabiulity of searching and representing multimedia learning resources in a 3-d virtual gaming environment. *IEEE Transactions on Instrumentation and Measurement*, 57(9):1830–1839, 2008.

[15] B. Entertainment. World of warcraft subscriber. Technical report, http://us.blizzard.com/en-us/company/press/pressreleases.html?id=2847881, Last accessed, August 2011.

[16] S. Fleck, F. Busch, P. Biber, and W. Straber. 3d surveillance a distributed network of smart cameras for real-time tracking and its visualization in 3d. In *Computer Vision and Pattern Recognition Workshop, 2006. CVPRW '06. Conference on*, pages 118 – 118, 2006.

[17] Gartner. 80% of active internet users will have a "second life" in the virtual world by the end of 2011. Technical report, http://www.gartner.com/it/page.jsp?id=503861, Last accessed, August 2011.

[18] A. Haans and W. IJsselsteijn. Mediated social touch: a review of current research and future directions. *Virtual Real.*, 9(2):149–159, 2006.

[19] S. Hossain, A. Rahman, and A. El Saddik. Interpersonal haptic communication in second life. In *Haptic Audio-Visual Environments and Games (HAVE), 2010 IEEE International Symposium on*, pages 1 –4, 2010.

[20] S. Hossain, A. Rahman, and A. El Saddik. Measurements of multi-modal approach to haptic interaction in second life inter-personal communication system. *to be published in IEEE Transactions on Instrumentation and Measurement*, 2011.

[21] R. L. Huard. *Plato's Political Philosophy: The Cave*. Algora Publishing, 2007.

[22] M. S. John, H. M. Oonk, and M. B. Cowen. Using two-dimensional and perspective views of terrain. 2000.

[23] S. King and J. Forlizzi. Slow messaging: intimate communication for couples living at a distance. In *DPPI '07: Proceedings of the 2007 conference on Designing pleasurable products and interfaces*, pages 451–454, New York, NY, USA, 2007.

[24] E. Kukula, M. Sutton, and S. Elliott. The human biometric-sensor interaction evaluation method: Biometric performance and usability measurements. *Instrumentation and Measurement, IEEE Transactions on*, 59(4):784 –791, 2010.

[25] S. Kumar, J. Chhugani, C. Kim, D. Kim, A. Nguyen, P. Dubey, C. Bienia, and Y. Kim. Second life and the new generation of virtual worlds. *Computer*, 41(9):46 –53, 2008.

[26] C.-A. La and P. Michiardi. Characterizing user mobility in second life. In *Proceedings of the first workshop on Online social networks*, WOSP '08, pages 79–84, New York, NY, USA, 2008.

[27] L. Lab. http://lindenlab.com. Technical report, Last accessed, August 2011.

[28] L. Lab. Uploading animations to second life. Technical report, http://community.secondlife.com/t5/English-Knowledge-Base/Uploading-assets/ta-p/700165, Last accessed, August 2011.

[29] P. Lamb. Artoolkit. Technical report, http://www.hitl.washington.edu/artoolkit/, Last accessed, August 2011.

[30] S. Life. Online user count. Technical report, http://secondlife.com/xmlhttp/secondlife.php, Last accessed, August 2011.

[31] R. Likert. A technique for the measurement of attitudes. *Archives of Psychology*, 140:1–55, 1932.

[32] LSL Portal. http://wiki.secondlife.com/wiki/lsl_portal. Technical report, Linden lab, Last accessed, August 2011.

[33] Microsoft Speech API. http://msdn.microsoft.com/en-us/library/ee125077(v=vs.85).aspx. Technical report, Last accessed, August 2011.

[34] S. O'Brien and F. Mueller. Holding hands over a distance: technology probes in an intimate, mobile context. In *OZCHI '06: Proceedings of the 18th Australia conference on Computer-Human Interaction*, pages 293–296, New York, NY, USA, 2006.

[35] A. Rahman, M. Hossain, J. Parra, and A. El Saddik. Motion-path based gesture interaction with smart home services. In *MM '09: Proceedings of the seventeen ACM international conference on Multimedia*, pages 761–764, New York, NY, USA, 2009.

[36] P. Rashidi and D. Cook. Keeping the intelligent environment resident in the loop. In *2008 IET 4th International Conference on Intelligent Environments*, pages 1 –9, 2008.

[37] A. Rovers and H. Van Essen. Him: a framework for haptic instant messaging. In *CHI '04: CHI '04 extended abstracts on Human factors in computing systems*, pages 1313–1316, New York, NY, USA, 2004.

[38] M. Rymaszewski, A. Wagner James, C. Ondrejka, R. Platel, and V. Gorden Sara. *Residents from around the world. Second life: The office guide.* Wiley Press., 2007.

[39] I. O. Sebe, J. Hu, S. You, and U. Neumann. 3d video surveillance with augmented virtual environments. In *First ACM SIGMM international workshop on Video surveillance*, IWVS '03, pages 107–112, New York, NY, USA, 2003.

[40] T. Seifried, C. Rendl, F. Perteneder, J. Leitner, M. Haller, D. Sakamoto, J. Kato, M. Inami, and S. D. Scott. Cristal, control of remotely interfaced systems using

touch-based actions in living spaces. In *ACM SIGGRAPH 2009 Emerging Technologies*, SIGGRAPH '09, pages 6:1–6:1, New York, NY, USA, 2009.

[41] S. Shirmohammadi and H. Nancy. Shared object manipulation with decorators in virtual environments. In *Distributed Simulation and Real-Time Applications, 2004. DS-RT 2004. Eighth IEEE International Symposium on*, pages 230 – 233, 2004.

[42] SLurl. Location-based linking in second life. Technical report, http://slurl.com, http://slurl.com/, Last accessed, August 2011.

[43] K. Stanney, M. Mollaghasemi, L. Reeves, R. Breaux, and D. Graeber. Usability engineering of virtual environments (ves): identifying multiple criteria that drive effective system design. *Int. J. Hum.-Comput. Stud.*, 58:447–481, April 2003.

[44] C. Thompson and F. Hagstrom. Modeling healthcare logistics in a virtual world. *Internet Computing, IEEE*, 12(5):100 –104, 2008.

[45] D. Tsetserukou, A. Neviarouskaya, H. Prendinger, M. Ishizuka, and S. Tachi. ifeel_im: innovative real-time communication system with rich emotional and haptic channels. In *CHI EA '10: Proceedings of the 28th of the international conference extended abstracts on Human factors in computing systems*, pages 3031–3036, New York, NY, USA, 2010. ACM.

[46] D. Tsetserukou, A. Neviarouskaya, H. Prendinger, N. Kawakami, M. Ishizuka, and S. Tachi. Enhancing mediated interpersonal communication through affective haptics. *Intelligent Technologies for Interactive Entertainment*, pages 246–251, 2009.

[47] W3C. Web Services Glossary. Technical report, http://www.w3.org/TR/ws-gloss, Last accessed, August 2011.

[48] W3C. XML-Encryption. Technical report, http://www.w3.org/TR/xmlenc-core, http://en.wikipedia.org/wiki/ XML_Encryption, Last accessed, August 2011.

[49] R. Wang and F. Quek. Touch and talk: contextualizing remote touch for affective interaction. In *TEI '10: Proceedings of the fourth international conference on Tangible, embedded, and embodied interaction*, pages 13–20, New York, NY, USA, 2010. ACM.

[50] C. Wickens, C. Liang, T. Prevett, and O. Olmos. Egocentric and exocentric displays for terminal area navigation. In *Human Factors and Ergonomics Society Annual Meeting Proceedings*, volume 38, pages 16–20. Human Factors and Ergonomics Society, 1994.

[51] Wikipedia. Radio frequency identification,. Technical report, http://en.wikipedia.org/wiki/Radio-frequency_identification, Last accessed, August 2011.

[52] X10. ActiveHome Pro SDK, http://www.activehomepro.com/sdk/index.html. Technical report, Last accessed, Aug 2011.

[53] B. Zhang, P.-L. P. Rau, and G. Salvendy. Design and evaluation of smart home user interface: effects of age, tasks and intelligence level. *Behaviour and Information Technology*, 28(3):239–249, 2009.

Appendix A

XML Schema

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="InteractionRules">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" name="hug">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="animationModules">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="UUID" type="xs:string" />
                    <xs:element name="animationBVH" type="xs:string" />
                    <xs:element name="animationPriority" type="xs:string" />
                    <xs:element name="animationLooped" type="xs:string" />
                    <xs:element name="animationSpeed" type="xs:unsignedByte" />
                    <xs:element name="animationDuration" type="xs:string" />
                    <xs:element name="animationScaleTo" type="xs:decimal" />
                  </xs:sequence></xs:complexType></xs:element>
              <xs:element name="tactileModules">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element maxOccurs="unbounded" name="module">
                      <xs:complexType mixed="true">
                        <xs:sequence minOccurs="0">
                          <xs:element name="highestIntensity" type="xs:unsignedByte" />
                          <xs:element name="lowestIntensity" type="xs:unsignedByte" />
                          <xs:element name="vibrationType" type="xs:string" />
                          <xs:element name="interactionTime" type="xs:unsignedShort" />
                          <xs:element name="numberOfRepetation" type="xs:unsignedByte" />
                        </xs:sequence>
                        <xs:attribute name="name" type="xs:string" use="required" />
                      </xs:complexType>
                    </xs:element></xs:sequence></xs:complexType></xs:element></xs:sequence>
              <xs:attribute name="userType" type="xs:string" use="required" />
              <xs:attribute name="hapticFeedback" type="xs:string" use="required" />
              <xs:attribute name="name" type="xs:string" use="required" />
            </xs:complexType>
          </xs:element>
          <xs:element name="touch" type="xs:string">...</xs:element>
          <xs:element name="tickle" type="xs:string">...</xs:element>
        </xs:sequence></xs:complexType></xs:element></xs:schema>
```

Figure A.1: XML schema for annotating haptic commands like hug, touch, tickle to the body parts of a Second Life avatar.

```xml
<?xml version="1.0" encoding="utf-8"?>
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="AnnotationRules">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="animationModules">
          <xs:complexType>
            <xs:sequence>
              <xs:element maxOccurs="unbounded" name="animationModule">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="UUID" type="xs:string" />
                    <xs:element name="animationBVH" type="xs:string" />
                    <xs:element name="animationLooped" type="xs:string" />
                    <xs:element maxOccurs="unbounded" name="animationSpeed" type="xs:string" />
                    <xs:element name="animationScaleTo" type="xs:unsignedByte" />
                  </xs:sequence>
                  <xs:attribute name="id" type="xs:string" use="required" />
                </xs:complexType>
              </xs:element></xs:sequence></xs:complexType></xs:element>
        <xs:element name="Appliances">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="Appliance">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="ApplianceType" type="xs:string" />
                    <xs:element name="DeviceType" type="xs:string" />
                    <xs:element name="PhysicalAddress" type="xs:string" />
                  </xs:sequence>
                  <xs:attribute name="name" type="xs:string" use="required" />
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType></xs:element></xs:sequence></xs:complexType></xs:element></xs:schema>
```

Figure A.2: XML schema for annotating a virtual 3D object in Second Life.