Faculdade de Engenharia da Universidade do Porto

Mixed-Reality Interactive Content Support on Second Life™

Jacinto João Maceda Brás

Tese submetida no âmbito do

Mestrado Integrado em Engenharia Electrotécnica e de Computadores

Major de Telecomunicações

Orientador: Ricardo Morla

Junho de 2008

Version 1.0 (July 30, 2008)

U.PORTO
FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

MIEEC - MESTRADO INTEGRADO EM ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES 2007/2008

A Dissertação intitulada

"Mixed-Reality Interactive Content Support on Second Life"

foi aprovada em provas realizadas 15/Julho/2008

o júri

Presidente Professor Doutor Artur Pimenta Alves
Professor Catedrático da Faculdade de Engenharia da Universidade do Porto

Professor Doutor Rui João Peixoto José
Professor Auxiliar da Escola de Engenharia da Universidade do Minho

Professor Doutor Ricardo Santos Morla
Professor Auxiliar da Faculdade de Engenharia da Universidade do Porto

O autor declara que a presente dissertação (ou relatório de projecto) é da sua exclusiva autoria e foi escrita sem qualquer apoio externo não explicitamente autorizado. Os resultados, ideias, parágrafos, ou outros extractos tomados de ou inspirados em trabalhos de outros autores, e demais referências bibliográficas usadas, são correctamente citados.

Autor - Jacinto João Maceda Brás

Faculdade de Engenharia da Universidade do Porto

Abstract

Nowadays Virtual Worlds and MMORPG offer users the advantage of easy programming and creation of highly interactive content but restrict the presentation and provided interactivity to the world they are in. On the other hand there is wide variety of Hardware in Real World to support interactive content and multimedia presentation but the content creation isn't has easy and the final result so interactive has it is on Virtual Worlds. The separation between Worlds and the way to provide support for something in the middle, in the "World" of Mixed-Reality , is the barrier blocking out users from the two Worlds to interact with the same contents and have the same advantages no matter in what World they are. This could be solved by taking advantage of the best in the two worlds and combine it to create a Mixed-Reality world(environment) with all the advantages of both and at the same time arrange ways to provide interaction to users in both worlds with this Interactive Contents in the Mixed-Reality environment context.

The objective of this project is to understand how support can be provided for interactive content on a Mixed-Reality environment. To help explore this issue, an architecture and mechanism were developed for playing interactive multimedia content in Second Life™ and in the real world. Mixed-Reality context interactivity is supported, with video on demand support on both worlds. System response to Mixed-Reality interactions bringing new functionalities in the Virtual and Real world, larger object variety and higher level of interactivity, to Real Users and a more Real, immersive, response to Virtual Users interactions by manifesting the result in a real fashion with supporting hardware are examples of benefits provided by our approach. After defining the goals of this project, we tested and validated an application architecture solution to use in Mixed-Reality Systems and a way to communicate between Real and Virtual World.

The contributions of the project are intended to make the experience of the Mixed-Reality User more fulfilling. Specifically provided support for interactive content on a Mixed-Reality environment composed by a Virtual World, Second Life™, and the Real World, a way to provide Audio and video streaming, including video on demand, in the same solution, a valid application architecture solution to deploy Mixed-Reality Systems and a possible low cost sensing solution providing the interactions for this kind of Systems, Mixed-Reality Systems.

Acknowledgments

I would like to thank to my supervisor, Ricardo Morla for all the help and especially for his patience, my family of course and my girlfriend Cristina for all the support and encouragement.

Jacinto

"What is real? How do you define real?
If you're talking about what you can feel, what you can smell, what you can taste,
see, then real is simply electronic signals interpreted by your brain."

Morpheus character from *"The Matrix"*

Contents

1 Introduction **1**
- 1.1 Context . . . 1
- 1.2 Problem . . . 2
- 1.3 Objectives . . . 2
- 1.4 Thesis Structure . . . 2

2 Related Work **3**
- 2.1 Mixed-Reality . . . 3
  - 2.1.1 Mixed-Reality Games . . . 3
  - 2.1.2 MXR/Augmented Reality/Head Mounted Displays . . . 3
  - 2.1.3 Mixed-Reality/Artistic performance . . . 4
- 2.2 Interactive Content . . . 4
  - 2.2.1 Presentations in Virtual Reality . . . 4
  - 2.2.2 Interactive Public Displays . . . 5
  - 2.2.3 Interactive computer graphics . . . 5
  - 2.2.4 Interactive TV . . . 5
  - 2.2.5 Interactive music . . . 6
  - 2.2.6 Interactive rooms . . . 6
  - 2.2.7 Interactive didactical Content . . . 6
  - 2.2.8 Interactive Art/Cinema . . . 6
  - 2.2.9 Interactive Multimedia development . . . 6
  - 2.2.10 Multimedia Support . . . 7
  - 2.2.11 Multimedia Context Adaptation . . . 7
  - 2.2.12 Visual Information Management . . . 7
  - 2.2.13 Video Streaming . . . 7
  - 2.2.14 Connectivity . . . 8
- 2.3 Mixed Reality/Interactive Content . . . 8
- 2.4 Virtual Worlds . . . 8
- 2.5 Interactive Content/Virtual Worlds . . . 9
- 2.6 Mixed Reality/Interactive Content/Virtual Worlds . . . 9

3 Technologies and tools **11**
- 3.1 Introduction . . . 11
- 3.2 Interactive Content/Multimedia . . . 11
  - 3.2.1 Conclusion . . . 13
- 3.3 Development Languages and Tools . . . 13
- 3.4 Communications . . . 14
  - 3.4.1 XML-RPC . . . 14

3.4.2 HTTP Requests . . . 14
3.5 Streaming Server . . . 15
3.5.1 Introduction . . . 15
3.5.2 Darwin Streaming Server . . . 15
3.5.3 Setup (installation) and deployment . . . 16
3.5.4 Compatible players . . . 18
3.5.5 Directories of DSS . . . 18
3.5.6 MP3 audio streaming . . . 18
3.5.7 MP4 video streaming . . . 24
3.5.8 Streaming Server and Firewalls . . . 24
3.5.8.1 Running Clients behind Firewalls . . . 24
3.5.8.2 Running Streaming Server behind Firewalls . . . 25
3.5.9 Bandwidth Considerations . . . 26
3.5.10 OS issues . . . 27
3.5.11 Conclusion . . . 27
3.6 Phidget RFID . . . 28
3.6.1 Introduction . . . 28
3.6.2 Setup . . . 29
3.6.3 Tags . . . 29
3.6.4 Library . . . 29
3.6.5 Conclusion . . . 29
3.7 Virtual World client . . . 29
3.7.1 Second Life and Firewalls . . . 30
3.8 Conclusion . . . 30

4 Requirements and Architecture 31
4.1 Introduction . . . 31
4.2 Requirements . . . 31
4.2.1 User point of view . . . 31
4.2.2 Application developer point of view . . . 32
4.3 Use Case . . . 32
4.3.1 System's Use Case diagram . . . 32
4.3.2 Real World diagrams . . . 34
4.3.2.1 Auto Play Package diagram . . . 35
4.3.2.2 Play Package diagram . . . 39
4.3.2.3 Playlist Management Package diagram . . . 42
4.3.3 Virtual World diagrams . . . 44
4.3.3.1 Auto Play Package diagram . . . 45
4.3.3.2 Play Package diagram . . . 48
4.3.3.3 Playlist Management Package diagram . . . 51
4.4 Domain Class Model . . . 53
4.4.1 Classes description . . . 53
4.4.1.1 IPLay . . . 53
4.4.1.2 Phidget RFID . . . 54
4.4.1.3 Tag . . . 54
4.4.1.4 User . . . 54
4.4.1.5 Userlist . . . 54
4.4.1.6 Music . . . 54
4.4.1.7 Video . . . 55

4.4.1.8 Audio Playlist . . . 55
4.4.1.9 Video Playlist . . . 55
4.4.1.10 SLClient . . . 55
4.4.1.11 SLServer . . . 56
4.4.1.12 Client . . . 56
4.4.1.13 Server . . . 56
4.4.1.14 Jukebox . . . 56
4.4.1.15 VirtualRFID . . . 56
4.4.1.16 VirtualTag . . . 56
4.4.1.17 Avatar . . . 56
4.5 Architecture . . . 57
4.5.1 Logical Architecture . . . 57
4.5.2 Physical Architecture . . . 58
4.5.2.1 Components Diagram . . . 58
4.5.2.2 Deployment Diagram . . . 59
4.6 Conclusion . . . 60

5 The Mixed-Reality Application 61
5.1 Introduction . . . 61
5.2 Virtual World (Second Life) Application . . . 62
5.2.1 Interface . . . 62
5.2.2 The Second Life "House" . . . 62
5.2.3 The Virtual Phidget . . . 63
5.2.4 Virtual Server . . . 63
5.2.5 The Video Player . . . 65
5.2.6 The main MXR application: the Jukebox . . . 66
5.2.7 Virtual Tags . . . 69
5.3 The iPlay Real World Support Application . . . 71
5.3.1 iPlay interface . . . 71
5.3.2 Design class diagrams . . . 71
5.3.3 Implementation Details . . . 73
5.3.3.1 IPlay Implementation Details . . . 73
5.3.3.2 Playlist and Userlist Implementation Details . . . 74
5.3.3.3 SLserver Implementation Details . . . 74
5.3.3.4 SLClient Implementation Details . . . 74
5.4 Conclusion . . . 75

6 Use Case Realization 77
6.1 Introduction . . . 77
6.2 The Mixed-Reality System at Work . . . 77
6.3 Conclusion . . . 83

7 Conclusion 85
7.1 Summary . . . 85
7.2 Results . . . 86
7.3 Future Work . . . 86

A LSL **89**
A.1 Jukebox Script . . . 89
A.2 Sensor Script . . . 95
A.3 Tag Script . . . 98
A.4 Client Script . . . 99
A.5 Server Script . . . 100
A.6 FreeView 1.2 Script . . . 102
A.7 Audio Playlist note card . . . 116
A.8 Video Playlist note card . . . 117

B Java code **119**
B.1 iPlay . . . 119
B.2 StateObj . . . 135
B.3 SLserver . . . 136
B.4 SLClient . . . 139
B.5 User . . . 140
B.6 Userlist . . . 142
B.7 Music . . . 144
B.8 Playlist . . . 145

References **150**

List of Figures

3.1 Server running the DSS's Ubuntu interface on VNC viewer . . . 19
3.2 Web Interface log in entry . . . 19
3.3 Web Interface main page . . . 20
3.4 Web Interface connected users' page . . . 20
3.5 Web Interface Playlists page . . . 21
3.6 Web Interface MP3 Playlist creation page . . . 21
3.7 Web Interface MP3 Playlist creation page . . . 22
3.8 Web Interface MP3 Playlist creation page . . . 22
3.9 Web Interface Playlists page . . . 23
3.10 DSS ports . . . 26
3.11 PhidgetRFID . . . 28

4.1 Use Case System diagram . . . 33
4.2 Use Case Real World Package diagram . . . 34
4.3 Use Case Real World Auto Play Package diagram . . . 35
4.4 Activity Real World Auto Play diagram . . . 38
4.5 Use Case Real World Play Package diagram . . . 39
4.6 Use Case Real World Playlist Management Package diagram . . . 42
4.7 Use Case Virtual World Package diagram . . . 44
4.8 Use Case Virtual World Auto Play Package diagram . . . 45
4.9 Activity Virtual World Auto Play diagram . . . 47
4.10 Use Case Virtual World Play Package diagram . . . 48
4.11 Use Case Virtual World Playlist Management Package diagram . . . 51
4.12 Conceptual Class Diagram . . . 53
4.13 Logical Architecture Diagram . . . 57
4.14 Components Diagram . . . 58
4.15 Deployment Diagram . . . 60

5.1 My Second Life "House" Overview . . . 62
5.2 The Virtual Phidget RFID and the Virtual Server(G5) . . . 63
5.3 Video Player . . . 65
5.4 Jukebox: the central application in Second Life . . . 66
5.5 Jukebox: Play . . . 68
5.6 The two RFID Virtual Tags . . . 69
5.7 Attaching the Tag to Avatar . . . 70
5.8 iPlay Interface (image captured in Windows Vista) . . . 71
5.9 Design Class Diagram . . . 72

6.1 The Mixed-Reality System . . . 77

6.2 Starting the program . . . 78
6.3 Enable information sending to Second Life . . . 78
6.4 Connecting Phidget RFID . . . 79
6.5 Phidget RFID connected . . . 79
6.6 Starting the Second Life Client . . . 80
6.7 Tag attached . . . 80
6.8 Virtual Phidget RFID connected . . . 81
6.9 Avatar with tag 1 detected . . . 81
6.10 Tag 2 Detected and Video Playing . . . 82
6.11 Tag 1 and Tag 2 detected . . . 82
6.12 Music playing in Firefox . . . 83

List of Tables

3.1 Compatible players 18

5.1 Decider method results 68

Chapter 1

Introduction

1.1 Context

This work leverages from three research areas: Interactive Content, Virtual Worlds, and Mixed Reality. Nowadays Interactive content is found everywhere from DVDs to TV game shows. Select the angle in a soccer game on SporTV [1] live transmissions, chose the movie you want to watch and the time when you want to watch it with video-on-demand, and move a character using the telephone keyboard in TV interactive games such as "Hugo" and "ARENA RADICAL" [2] from Alcatel and Sic Radical. These are all examples of how you can interact with multimedia content you see on TV. With the advent of digital terrestrial TV the options will soon be much more.

A virtual world is a computer generated environment where users can interact with each other's using avatars. In generic, online, virtual worlds such as Second Life™ [3] the interactions are more social oriented and can mimic everyday interactions of the real world. You can customize your avatar, buy a house and even get a job. Then you have the massively multiplayer online role-playing games (MMORPG) such as EverQuest, Ultima Online, Lineage, Guild Wars and World of Warcraft where like in Second Life™ you move in a virtual world but this time you have quests to play and you can do it cooperatively with other online gamers.

Mixed-Reality consists of merging of real and virtual worlds where physical and digital objects co-exist and interact in real time. This is what systems such as Sony® Playstation® EyeToy® [4] provide. EyeToy® is composed of a USB camera with a built-in microphone, which allows players to interact with games using image processing-detected motion, color or sound captured.The new music video "Clumsy" from Fergie [5] now playing on MTV is a good example of mixed reality the final effect is similar to the holograms from Star Wars but looks more real.

1.2 Problem

Nowadays Virtual Worlds and MMORPG offer users the advantage of easy programming and creation of highly interactive content but restrict the presentation and provided interactivity to the world they are in. On the other hand there is wide variety of Hardware in Real World to support interactive content and multimedia presentation but the content creation isn't has easy and the final result so interactive has it is on Virtual Worlds. The separation between Worlds and the way to provide support for something in the middle, in the "World" of Mixed-Reality , is the barrier blocking out users from the two Worlds to interact with the same contents and have the same advantages no matter in what World they are.

1.3 Objectives

The objective of this project is to understand how we can provide support for interactive content on a Mixed-Reality environment composed by a generic massive multiuser online Virtual World, such as Second Life™, and the Real World, take advantage of the best in the two worlds and combine it to create a Mixed-Reality world(environment) with all the advantages of both and at the same time arrange ways to provide interaction to users in both worlds with this Interactive Contents in the Mixed-Reality environment context bringing the best of Virtual to Real Users and best of Real to Virtual Users.

1.4 Thesis Structure

Beyond the Introduction this thesis has 6 more chapters.

On chapter 2, Related Work is presented organized by the research areas they approach.

On chapter 3, Technologies and Tools, the state of art on Multimedia, the technologies and the tools used to develop the system are described.

On chapter 4, Requirements and Architecture, are presented the initial expectations for what the system could do and the architecture to support it.

On chapter 5, Mixed-Reality application, the implementation of the system is addressed first the Real World then the Real and the main aspects and details of the written code are described.

On chapter 6, Use Case Realization, the architecture is validated by testing the system response to the inputs and see if the output is the predicted.

On chapter 7, Conclusion, the summary, the results achieved and future work are addressed.

Chapter 2

Related Work

2.1 Mixed-Reality

2.1.1 Mixed-Reality Games

A Mixed-Reality (MXR) example is "Can You See Me Now"(CYSMN) a location–based game [6]developed by Mixed-Reality Laboratory and Blast Theory that consists of a real time game where online gamers moving in a virtual model of a real city have to run from the runners that are trying to catch the online players but are moving in the real city. This is done with the deploy of a wireless network with access points distributed through the city trying to cover the complete game area and a GPS system to get the runner coordinates which then are sent to the online server to cross the real coordinates with the virtual online players ones. If the runner gets within a certain distance from the player the player is caught. It can relate to my work in the interaction between the game and the real world and a sense of presence in both worlds by users but in this case the Virtual World is just a city map, much more restrictive than other Virtual Worlds such as Second Life™.

ARQuake[7] is an attempt to bring the First Person Shooter Quake into the streets. The Real World is the Map of the game and the point of view of the game is determined by the head position and orientation of the player. Information is displayed as augmented reality with the help of a head-mounted see-through display. The interface is complemented by a hand-held button device. Another classic game Pacman as received a similar treatment. Human Pacman[8], like ARQuake, is played in the real world and additional information is also displayed as augmented reality with a HMD but in this case the ghosts are humans like the runners of CYSMN and the gameplay is also similar to CYSMN.

2.1.2 MXR/Augmented Reality/Head Mounted Displays

Augmented reality refers to the real-time insertion of computer-generated graphical content into a real scene. 3D Live[9] inserts three-dimensional dynamic graphical objects into

the real world giving the idea of telepresence using 15 cameras to capture the image and a head mounted display with a small security camera attached to the front so that the user sees the real world but modified with the 3D image of the collaborator in the scene.This is based in image and motion capture and although the subject still MXR this goes out of the focus of my project.

Another example is the research being made by BBC R&D and BBC Creative R&D[10] in the area of augmented reality and the possibility of including it in live broadcasting of interactive content. In this case like in my project the content is interactive and the subject Mixed-Reality so it might be a good reference but it's not what I'm going to do.

[11]Describes the use immersive projection displays in telecommunications using video avatars that are superimposed on the shared virtual world. The final result is impressively similar to the holographic communication system from the movie Star Wars with the difference that the video avatars are not superimposed on the real world but on virtual one. This is an interesting subject but apart from being about Mixed-Reality there are no more relations to my work.

2.1.3 Mixed-Reality/Artistic performance

"Murmuring Fields"[12][13][14] is an artistic installation based on mixed reality and shared environment. It's composed of a virtual and a real stage. In the real stage the movement of the performers is captured with computer vision and transformed into sound according to the speed of movement and direction. The interface is the space and full-body movement instead of a physical device. Movement triggers sounds presents in the virtual world, sounds that then can be heard in the real space and seen in the virtual world as a trace-avatar (a shadow like representation). The sounds are triggered also by virtually touching sound objects in the virtual world. e-MUSE[12][13][14] (or eMUSE) system (electronic multi user stage environment) is the supporting system, the underlying platform, to "Murmuring Fields" installation and can be used for rapid prototyping of Mixed Reality architecture. The Murmuring Fields and eMUSE projects are related to mine but with a much more immersive Mixed-Reality approach.

2.2 Interactive Content

2.2.1 Presentations in Virtual Reality

[15]Describes concepts on how to transform normal text and images presentations in interactive content 3D presentations to be viewed as a slideshow in a virtual environment. It has interactive content but does not directly relate to my work.

2.2.2 Interactive Public Displays

A good example of ubiquitous computing and interactive content is the e–campus system from Lancaster University [16][17]. This system includes a giant screens net on which is scaled a series of contents with which users may interact through cell phone or web cams with motion detection. Part of e–campus deployment problem is the placement of ubicomputing [18] which needs to be where it can be useful to users and can catch their attention.
ContentCascade[19] is another project on the subject of interactive public displays allowing users to download summary information about what's passing in the display to their portable devices for later review.
Another similar project[20] addresses the subject of advertising based on presence detection of Bluetooth devices as a way of ensuring an advertisement receives maximum exposure.
These are all similar to my project in the way that interactive content is displayed.

2.2.3 Interactive computer graphics

[21]Presents a system to control computer graphics trough a computer vision algorithm. For instance with the movement and rotation of your hand you control a plane in a game. It's a good example of interactive content but not directly related to my work.

2.2.4 Interactive TV

MITv[22] addresses the subject of interactive TV proposing a solution based on satellite DVB-S, AD-HOC infrastructures, xDSL and cable DVB-T. Based on IP Multicast the system is capable of offering regular broadcasting TV and the Interactive Content associated.
Another approach to the same subject[23] adding interactive applications to the broadcast stream making the content delivery over IP. Multimedia Home Platform set-top box running a Java virtual machine is needed by the end user to run the downloaded applications.
Simutainment[24] developed by BBC R&D Team presents a way of converting Documentaries into a 3D Interactive form making them look like video games. The presented example is based in the Jurassic era and allows the user to explore a virtual 3D world and see the dinosaurs and plants from that era as well as hearing the narration.
[25]This presents a model to structure and to organize documents describing Interactive TV programs and its related media objects. It gives support to metadata(contextual information) making it possible to present information about in-frame video objects.
Interactive TV Content goes beyond the objectives of my work.

2.2.5 Interactive music

RadioDrum[26] is a content-aware music browser which browses not based on metadata from the file but based on the tempo and beat strength of the music using two drum sticks and the Radio Drum/Baton to give the system the reference to search.
Musescape[27] is similar to RadioDrum, it's also a content-aware music browser, but the search is based on the signal properties of the music.
Both of them address the subject of interactive content but are not in any other way related to my work.

2.2.6 Interactive rooms

iRoom[28] is a project focusing on creation of an interactive workspace with touch sensitive white-board displays, a 183cm display, PDAs and laptops all communicating with each other. This could be helpful as example when deploying the communication part of my project.

2.2.7 Interactive didactical Content

StoryToy[29] is an interactive storytelling farm for children with audio narrative. With stuffed animals with wireless sensor pods inside the interaction with them triggers the "game" engine that sends the audio feedback. [30]Discusses approaches to build and modify on-line interactive didactical virtual experiments.
Interactive didactical content does not directly relate to my work.

2.2.8 Interactive Art/Cinema

In recent movies such as "Beowulf" you can see IMAX a new 3D method using different glasses from the classic red and blue lens. The VR theater[31] combines the advantages of VR and IMAX theater but enables multi-user interaction with the movie by using an interaction keypad.
The Very Nervous System uses a video camera to detect position and movement. The computer then maps the movement information into sound. This can be used in dance performances[32].
[33]Examines art as a general approach focusing on the process of interaction as opposed to generating a final product.
The subject of interactivity in art and cinema is mentioned here merely to state the new application fields of interactiveness. This subject is not related to my work.

2.2.9 Interactive Multimedia development

[34]States a method for developing interactive multimedia through the modeling of their semantic content i.e. metadata.

AgentSheets[35]is an agentbased simulation-authoring tool that allows end-users to build interactive simulations and publish them as Java applets on the web.
I'm most likely being using existing interactive multimedia in my project so this specific approach goes beyond the goals of my project.

2.2.10 Multimedia Support

[36]Presents a middleware and an application framework, based on it, to support/control a multimedia media center allowing different portable devices to exploit the capabilities of a nearby station. This is similar to Microsoft Media Center but more extensible and flexible because it's not based on closed hardware and software like Media Center. This relates to my work because can give some ideas on how to deploy my multimedia content supporting application.

2.2.11 Multimedia Context Adaptation

URICA[37] is an application to adapt content to devices based on user preferences and usage semantics improving bandwidth usage and reducing latency. This might be useful to improve network QoS in my work.

2.2.12 Visual Information Management

MetaSEEk[38] is a internet search engine to find multimedia files based on meta-data and semantic description of the files i.e. instead of searching for a video file called superman it searches for video files where superman appears. Although not directly related to my work it would be an excellent extent to any interactive Video/Media on demand application and also to my project.

2.2.13 Video Streaming

Streaming is crucial to my work [39] describes a method for MPEG-2 video streaming with full interactive functions included on the stream. This is achieved by storing different encoded versions of the video on the server one for normal playback the others for interactive functions such as fast forward.
Another streaming method is described by [40] but in this case based on H.264 codec and directed to streaming on PDAs. Although it addresses mobile video surveillance being based on low-latency video streaming and being open-source makes it an interesting proposal to take into account. Both these are interesting and if not directly applicable in my project can give me some ideas and guidelines.

2.2.14 Connectivity

SyncTap[41] system uses synchronous user operation for establishing spontaneous network connections between digital devices. In other words by pressing the sync button in two devices, first in one then in the other, a connection is established between the two just like Bluetooth connections. Not being directly related to my work it addresses the connectivity issue important to my project.

2.3 Mixed Reality/Interactive Content

Mixed Reality Book[42] makes the user/reader part of the story. The user reads the book through a handheld display and a computer screen in the back shows what he is watching. It allows us to appreciate multi-sensory content by keeping the physical book. Just like the Mixed Reality Book there is also the MagicBook[43] similar to it in every aspect.

i2TV[44] is an online Mixed Reality TV. The Mixed Reality Stage based on e-MUSE[12] brings together several participants in physical space with participants from the Internet.

2.4 Virtual Worlds

There is a project in development concerning Virtual Worlds named "Second Earth"™ [45] combining Second Life and Google Earth © [46] which will provide a more realistic and therefore immersive Virtual World. In my project I will use an already implemented Virtual World so although this project it's also about Virtual Worlds it goes in a different direction than mine.
Concerning Second Life the project "VU Campus"[47] consisted on building a virtual campus, in this case the VU University of Amsterdam, in this Virtual World. They make references to LSL and the recreation of real objects in virtual form. Although their content is much more extensive this can be helpful in the creation of the virtual content in my project. Another interesting subject is the recreation of medical and health education facilities in Second Life.
In *"Second Life: an overview of the potential of 3-D virtual worlds in medical and health education"*[48] two examples are used as "case studies" "HealthInfo Island" and "VNEC" to see the pedagogical potentials of Virtual Worlds. This does not address the goals of my project but is interesting to see the impact and importance Virtual Worlds such as Second Life has nowadays. Rudy P. Darken and John L. Sibert from The George Washington University developed tools to navigate in Virtual Worlds[49] based on real ones such as maps. User position in the Virtual World is important but the navigation issue is irrelevant to my work.

2.5 Interactive Content/Virtual Worlds

Virtual Video Gallery[50] is a Media on Demand system browsable through a virtual world. The user "walks" in the virtual gallery and chooses the media to watch. A preview option is also available. This can relate to my project differing in the fact that I have a wider virtual world instead of a virtual gallery.

2.6 Mixed Reality/Interactive Content/Virtual Worlds

[51]Describes a model for Mixed Reality MMORPGs that bases the narrative on the model of the user playing the game. The gamer interacts with the game trough normal actions and even emotions and the game also interacts with user sending SMSs to user for example and the narrative evolutes based on the those interactions. This definitely can relate to my work as it approaches the same major research fields.

Chapter 3

Technologies and tools

3.1 Introduction

In this chapter will be discussed the technologies used in the project, some of their advantages and disadvantages and the reason they were chosen. Also the tools used and the reason to use them will be briefly described. This chapter can be considered as part of the architecture, the Technological Architecture.

3.2 Interactive Content/Multimedia

An audio/video codec is a compression (lossy therefore irreversible) algorithm based on standards, like JPEG for images, which enables us to compress a video, reducing its size, without losing much quality. There are hundreds of codecs for audio and for video but I will just refer the ones I use more often and that I know best.

Audio codecs:

- MP3: MPEG-1 Audio Layer 3 audio compressor(codec). Capable of providing great reduction in audio files size with almost imperceptible losses in quality it revolutionized the music industry and started, back in the 90's, a war that still is going on today between consumers and Record Companies (much because of napster and the downloads for free over the internet). It changed the way we listen to music. It is also an audio format;

- AC3: Dolby Digital® audio codec. Supports up to 6 audio channels and up to 48 KHz sample-rates and it's used in theaters. It is also used in DVDs, HD-DVDs, Sony's Blu-Ray and Digital TV. If quality is what you're after this is the codec to use;

- WMA: Microsoft® response to MP3 an Real audio and similar to them. It is also an audio format;

- lameMP3: Similar to mp3 in every aspects providing better quality/size ratio;
- RM: Real® Media format audio codec. It's very used by radios for streaming live over the internet. It is also an audio format;
- Sony®'s atrac3: used on the primarily in the miniDisk is supported by Sony multimedia players (including the PSP) achieving a better quality/size ratio than mp3 crucial in portable media players. The only problem is that the support is restricted to Sony® products.

Video codecs :

- DivX® (MPEG4 based):lossy MPEG-4 Part 2 compression created by DivX, Inc. It started back in 98 as DivX ;-) 3.11 Alpha a hacked version of the Microsoft MPEG-4 and was very popular (still is) in the DVD-ripping(which is legal if you own the original) community. It allowed MPEG4-V3 to be saved in avi format instead of asf from Microsoft. Very good Quality/Size Ratio and very useful. It became a brand and is supported by some DVD players;
- Xvid (MPEG4 based): Open source version of DivX, very similar in every aspect with the advantage off unlike DivX, only supported (officially) by Microsoft Windows, being cross-platform;
- H.264: A.k.a. MPEG4 Part10 was created to make possibly to achieve the same quality of other MPEG4 codecs with half or less the bit rate therefore making possible higher resolutions and quality while keeping the file at a reasonable size. While HDVD and Blu-Ray battle continuous you can already enjoy High Definition video in your PC tanks to this.

Next will be referred the most common audio video formats with a brief description and capabilities for streaming.

Audio/Video Formats:

- Macromedia® flash video (flv): it's a very popular format on the internet (YouTube), its fast but the quality is not very good;
- Microsoft® wmv,wma, asx(asf codec): WMV was originally designed for Internet streaming applications as a competitor to RealVideo. It's fast and has good quality;
- AVI: Audio Video Interleaved was introduced by Microsoft® as a Video for Windows® technology. It's a multimedia container format and it supports an enormous quantity of codecs for video and for audio. Very useful for Home movies and conversions (DVD to avi) but not vey used for streaming;
- Real® rm,rsx: Real® Media container format. Current version is RealAudio 10. rsx is the streaming format but is equal to rm; Very good for streaming;

- Apple® QuickTime: qt,mov(based on MPEG4 standard): From Apple computers inc. the version 7 is based on MPEG4 Part10(H.264). Is used on Apple Trailers and in almost all trailers from upcoming movies that you can see on the internet. It's relatively fast and has very good quality;
- mpeg/mpg: format based on the standards MPEG and MPEG2. Mpeg2 is used on DVD movies. Both are not very suitable for streaming;
- DivX®: more known as a codec is also a format and used on the new YouTube like site dixstage6. It's relatively fast and has very good quality;
- MP4(MPEG-4 Part 14): It's an extension from H.264 based on the Apple® QuickTime format and allows streaming over the Internet. It's kind of the new MP3 and very popular in portable multimedia players such as Apple's iPod® and Sony®'s PSP. Also it is possible if you have a Sony®'s PS3 and a PSP to stream video in this format (and also in DivX) from the PS3 to the PSP;
- 3GP(MPEG-4 Part 12): Used in 3G mobile phones and PDAs. It's fast but with poor quality.

3.2.1 Conclusion

The format used for video streaming was MP4 which using H.264 for video and AC3 for audio supports High Definition (HD) video @ 1080p and Dolby Digital 48 KHz 6 channels audio in a video file with one hour length that fits on a DVD9. On the other hand if we privilege speed over quality since H.264 can output video with half the bit-rate of DivX for the approximately the same quality, this is an excellent codec for streaming video. For audio we used MP3 (with MP3 or lameMP3 codec) to ensure compatibility.

3.3 Development Languages and Tools

Possible languages to use were C#, C++, Java, LSL, LibSecondlife library functions (C#). LSL and Java were used.

LSL, Linden Scripting Language, it's a programming language with C-like syntax that is used to develop scripts in Second Life. For LSL script editing I used Notepad++ for Windows®. An offline editor/compiler is also available just google for it.

Developed by Sun Microsystems® and released in 1995 as a core component of Sun's Java platform this programming language syntax is very similar to C and C++ but simpler (more high level). Unlike C/C++ compiled programs Java programs (jars) run on a java virtual machine (JVM and JRE when running) making then OS independent. Editing and compiling was made with Eclipse IDE, an open source project mostly known for its Java IDE, with some plugin extensions for build and editing the interface (Visual Editor, EMF and GEF) and create to create the deployment class diagram(Omondo).

To draw the UML diagrams MS Visio® was used.

3.4 Communications

An option to communicate with Second Life Servers is LSL http Requests other is to use Libsecondlife to build a 3rd party client. XML-RPC is only available in one way, outside to inside of SL, at least for now. So XML-RPC was used to send requests to SL and HTTP Requests to do the same but from SL to the "Real World".

3.4.1 XML-RPC

XML-RPC it's a spec and a set of implementations (Java, C++,C#...) that allow software running on disparate operating systems, running in different environments to make procedure calls, that means you call a procedure (function) on a different machine, over the Internet using XML as a common language(encoding) and HTTP as the transport to transmit data. XMLRPC is designed to be as simple as possible, while allowing complex data structures(arrays, strings, structures...) to be transmitted, processed and returned. In other words it allows you to run a remote process or function in a different machine with only XML in common and return the results of that calling. So you can run a function from a different running program in another programming language and running in a different machine.

The Java package org.apache.xmlrpc provides classes to implement an XML-RPC client and an XML-RPC server. The package can be found at http://ws.apache.org/xmlrpc/. A copy of the package is under the name cis69mc.jar on Blackboard which was the one used in the application.

3.4.2 HTTP Requests

Hypertext Transfer Protocol (HTTP) is a communications protocol for the transfer of information on the intranet and the World Wide Web. It is a request/response standard between a client and a server. Typically, an HTTP client initiates a request establishing a Transmission Control Protocol (TCP) connection to a particular port on a host (port 80 by default). An HTTP server listening on that port waits for the client to send a request message. Upon receiving the request, the server sends back a status line, such as "HTTP/1.1 200 OK", and a message of its own, the body of which is perhaps the requested file, an error message, or some other information. An HTTP request consists of a request method, a request URL, header fields, and a body. HTTP 1.1 defines the following request methods:

- GET: Retrieves the resource identified by the request URL
- HEAD: Returns the headers identified by the request URL

- POST: Sends data of unlimited length to the Web server
- PUT: Stores a resource under the request URL
- DELETE: Removes the resource identified by the request URL
- OPTIONS: Returns the HTTP methods the server supports
- TRACE: Returns the header fields sent with the TRACE request
- CONNECT: Converts the request connection to a transparent TCP/IP tunnel, usually to facilitate SSL-encrypted communication (HTTPS)

HTTP 1.0 includes only the GET, HEAD, and POST methods.

3.5 Streaming Server

3.5.1 Introduction

The objective being to stream to Second Life makes the choosing of the streaming server limited. Since the plugin used on Second Life to play video is from Apple's QuickTime Player the Server would have to be capable of streaming either quick time movie or mp4 (MPEG-4 H264) format otherwise video would not play. The choice of format was for mp4 because even if both formats use h264 codec, for mp4 there are lots of open source encoders while for quick time you must have QuickTime Player Pro, which is shipped with Mac OS X Server, or buy it. So the obvious choice of the server, which as it was said must be compatible with Quick Time player, would be Quick time Streaming server but then again it would be preferable to use open source software and fortunately Apple has a Open Source version from their Server the Darwin Streaming Server.

3.5.2 Darwin Streaming Server

Based on the same code base as QuickTime Streaming Server (QTSS), Darwin Streaming Server(DSS) is the open source version of this technology which delivers hinted QuickTime (.mov), hinted MPEG-4 (.mp4), and hinted 3GPP (.3gp) files in real time over the Internet and mobile networks via the Real-time Transport Protocol/Real-Time Streaming Protocol (RTP/RTSP). It can also deliver MP3 files via Icecast-compatible protocols. It delivers both video on demand (VoD) and, when combined with broadcasting software, live streams. It is perfect for serving live events over the web, when partnered with broadcaster software, or for creating a 24x7 video or radio station with the included Playlist Broadcaster, or delivering long-form media. It is also perfect for those concerned with customers downloading files locally: real time streaming means the data is "consumed" as it is delivered, leaving no file to play back locally.

It is intended for developers who need to stream QuickTime and MPEG-4 media on alternative platforms such as Windows, Linux, and Solaris, or those developers who need to extend and/or modify the existing streaming server code to fit their needs. Both DSS and QTSS are built on a core server that provides state of the art quality of service features with Skip protection and Instant-On, and support for the latest digital media standards, MPEG-4 and 3GPP.

3.5.3 Setup (installation) and deployment

1. Make sure that you have: libc6-dev linux-libc-dev gcc-3.3 g++-3.3, if not:

```
sudo apt-get install libc6-dev linux-libc-dev gcc-3.3 g++-3.3
```

2. Download the source code of DSS-Source-,but before you should create an account in http://www.apple/developer to download DarwinStreamingSrvr5.5.5-Source.

3. Extract the file directory:

```
tar xzf DarwinStreamingSrvr5.5.5-Source.tar.gz DarwinStreamingSrvr5.5.5-
    Source
```

4. Create a group and user for Darwin:

```
sudo addgroup --system qtss
sudo adduser -system -no-create-home -ingroup qtss qtss
```

   Not necessary, but if you want change password to qtss (user) for example "xx"

```
sudo passwd qtss
```

5. Enter to "DarwinStreamingSrvr5.5.5-Source" directory and type:

```
./Buildit install
```

6. With this command starts to compile DSS and creates a new directory called "DarwinStreamingSrvr5.5.5-Linux"

7. Enter to "DarwinStreamingSrvr5.5.5-Linux" and type:

```
sudo ./Install
```

8. The command will show:

```
Installing Darwin Streaming Server

...

Installation Complete
```

```
Darwin Streaming Server Setup

In order to administer the Darwin Streaming Server you must create an
    administrator user
[Note: The administrator user name cannot contain spaces, or single or
    double quote characters, and cannot be more than 255 characters long
    ].
Please enter a new administrator user name: ee98179

You must also enter a password for the administrator user
[Note: The administrator password cannot contain spaces, or quotes,
    either single or double, and cannot be more than 80 characters long
    ].

Please enter a new administrator Password:
Re-enter the new administrator password:
Adding userName ee98179
Setup Complete!
```

9. Open your favorite browser and type:

```
http://localhost:1220/
```

   If DSS is down! Run the DSS web server:

```
sudo /usr/local/sbin/streamingadminserver.pl
```

10. To listen an mp3 list created (if you haven't already you must create one first):

```
http://localhost:8000/mylist (mp3 Broadcast)
```

11. To watch a movie:

```
rtsp://localhost/sample_100kbit.mp4
```

3.5.4 Compatible players

Not all players support URL playback of mp3 and rtsp. The following table shows some of the most common players and their ability to play back streaming content (X: doesn't work , OK: works).

Player	mp3	rtsp
mplayer (classic)	X	OK
wmplayer (*)	OK	OK
QuickTime Player (**)	OK	OK
vlc	OK	OK
helix	X	X
ibm applet	OK	X

Table 3.1: Compatible players

(*)Windows Media Player 10 and above
(**)including browsers that use QuickTime plugin such as Firefox

3.5.5 Directories of DSS

The following box presents an overview of the most important files and directories of the Darwin server:

```
/usr/local/sbin/Darwin Streaming Server - Server Software
/usr/local/sbin/streamingadminserver.pl - Web Frontend
/etc/streaming - Configuration Dir
/etc/streaming/streamingserver.xml - Configuration File Server
/var/streaming/logs - Logs
/usr/local/movies - Default directory for video files
```

3.5.6 MP3 audio streaming

With the Darwin server can not only MPEG-4 videos, but also MP3 audio files stream. You can create MP3 files, for example, ripping your CDs with your playback or cd burning software (Windows Media Player and Nero Burning Rom have this feature by default). If you want a free Ripper google for Audiograbber a free open source CD audio ripper. Then copy your mp3 files to the streaming directory of DSS and create your playlists, which, thanks to the existing web interface can be made relatively quickly.

Just follow these steps to see some of the web interface pages brief description and how to create your own mp3 playlists:

1. If the interface is not running you must start it with the command *sudo /usr/local/s-bin/streamingadminserver.pl*. This is the interface of our server where the DSS is installed, controlled remotely with VNC viewer, and where you can see, in the shell, the command that started our web interface. As you can see after entering in super user mode you just need to type *streamingadminserver.pl* because the installation in Ubuntu automatically adds DSS directories to the system path.

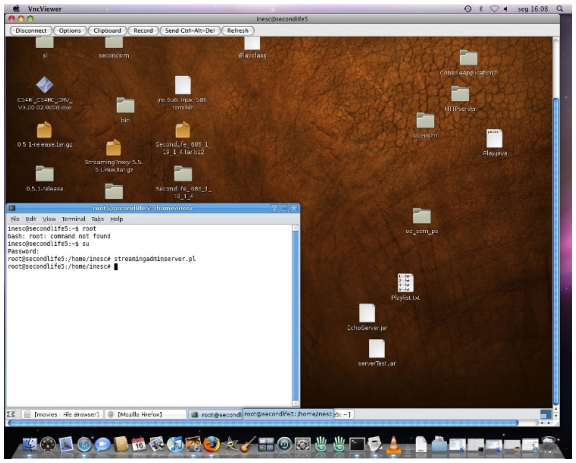

Figure 3.1: Server running the DSS's Ubuntu interface on VNC viewer

2. Open your browser, type in the server URL *http://¡Server-IP¿:1220/* and your log in information and press enter.

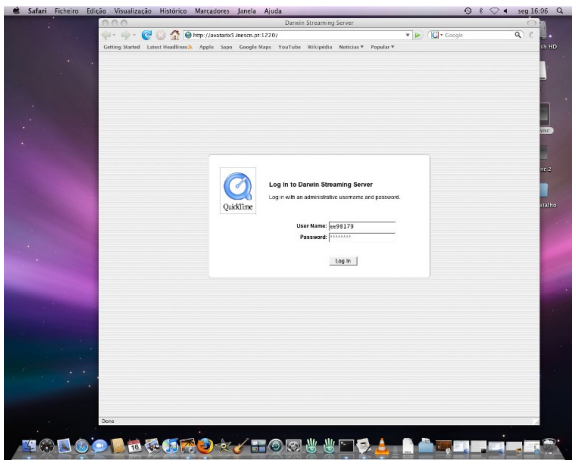

Figure 3.2: Web Interface log in entry

3. After logging in the main page of the web interface is displayed to you, were you can see some information relative to the server.

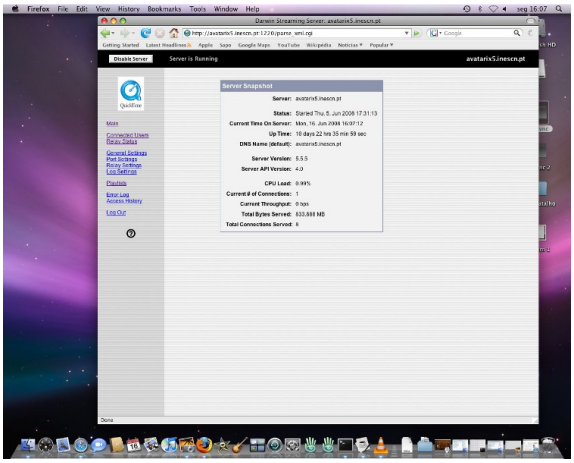

Figure 3.3: Web Interface main page

4. In this page you can see information regarding the connected users and some statistical information such as packets lost.

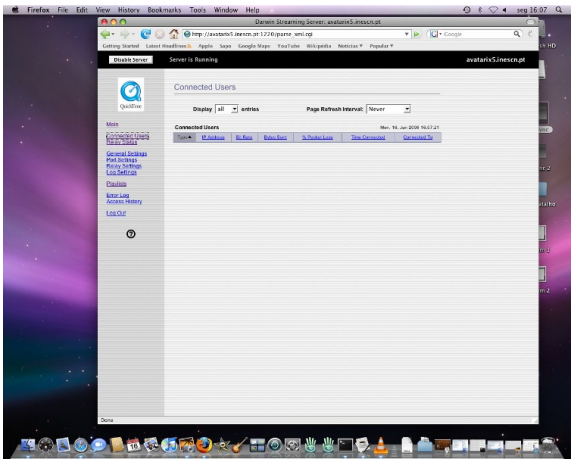

Figure 3.4: Web Interface connected users' page

5. The Playlists page shows you the current playlists and their status (playing or stopped). You can create a new mp3 playlist, a new media playlist or after selecting one edit or delete it.

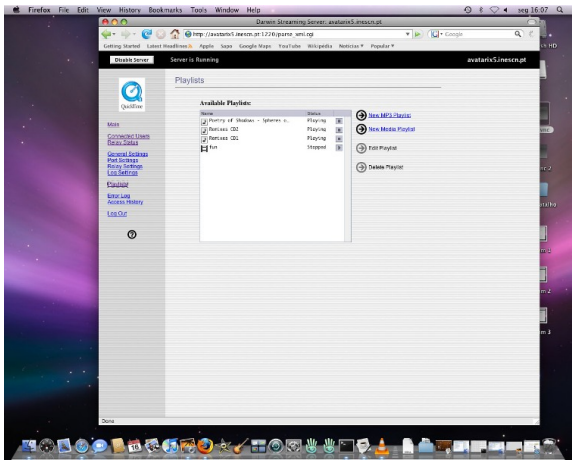

Figure 3.5: Web Interface Playlists page

6. By clicking on "New MP3 Playlist" button this page will show up.

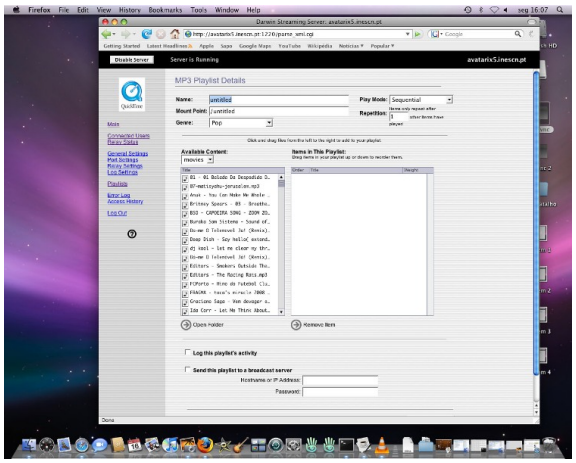

Figure 3.6: Web Interface MP3 Playlist creation page

7. Just drag and drop the mp3 files or directories from the left to the right list box and type in the mounting point and name. The rest of the options are optional and you can just leave them with their default values. Nevertheless it's better to change the "Play Mode" to "Sequential Looped" this way the files will play from the first to the last and then repeat.

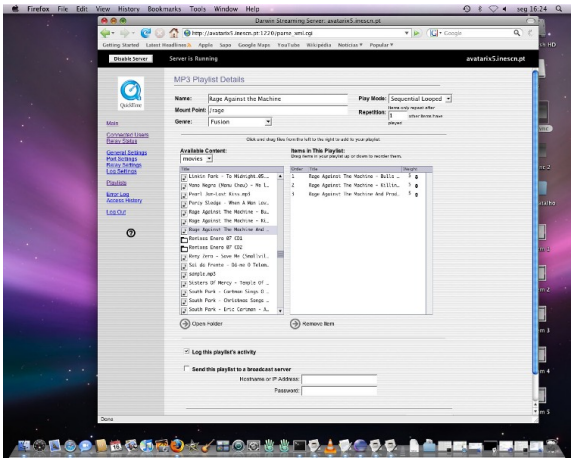

Figure 3.7: Web Interface MP3 Playlist creation page

8. After adding the mp3 files and typed the required information just click save at the bottom of the page.

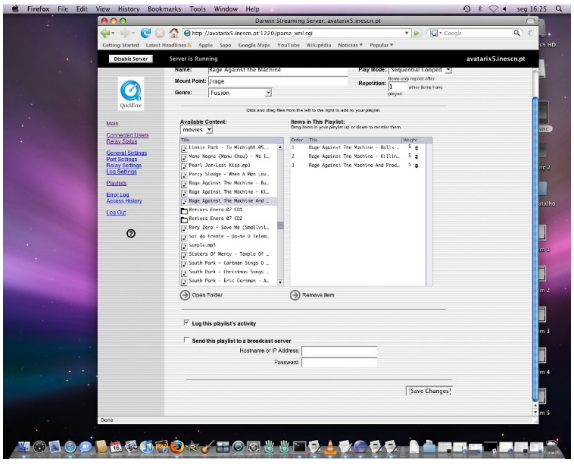

Figure 3.8: Web Interface MP3 Playlist creation page

9. Now in the Playlists page you can see your newly added playlist. Now to begin streaming the new playlist just click the play button next to the status information.

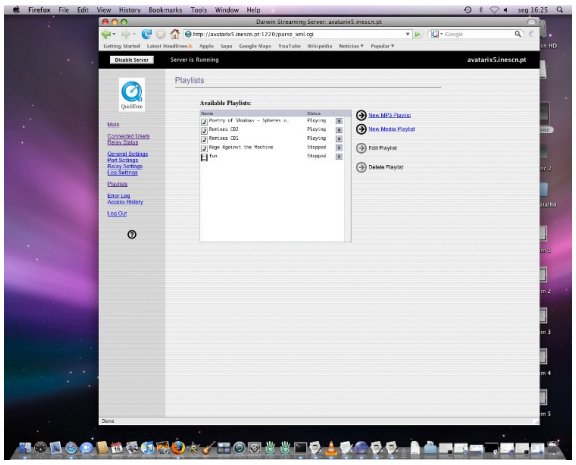

Figure 3.9: Web Interface Playlists page

To play a playlist in a MP3 player or browser open the playlist URL address which will have the following format:

```
http://<Server-IP>:8000/mountingpoint
```

3.5.7 MP4 video streaming

In order to use the server as a streaming video-on-demand server videos must be in MPEG-4 format (file extension .mp4). Using free video editing tools or DVD ripping tools you can create your own mp4 files. A good free converter it's the PSPvideo9 which is free but since it was designed to convert videos for the Sony PSP the resulting quality and resolution are not the best. Videos in QuickTime format (.mov) could also be used, but are probably less interesting.

However, before an MPEG-4 video with the Darwin server can be streamed must have known in advance hint tracks in the data stream added. The hint tracks tell the server exactly how to package the media data for the network. They are required to fast forward and rewind the video. For inserting the hint tracks use Tool MP4Box in the package gpac (multiverse, [2]) for Ubuntu.

The syntax for the command call for inserting the hint tracks reads:

```
MP4Box -hint dateiname.mp4
```

With the option -unhint inserted information may also be removed.

Alternatively, you can hint the tracks with the tool mp4creator from the MPEG4IP project.

After inserting the hint tracks, copy or move the video file into the video directory of the Darwin server, immediately thereafter, the video may be accessed via the web interface to be included in a media playlist (which are created similarly to MP3 ones) or by opening the video URL directly in a player or browser like this:

```
rtsp://<Server-IP>/<videoname>.mp4
```

3.5.8 Streaming Server and Firewalls

3.5.8.1 Running Clients behind Firewalls

The streaming server uses the IETF RTSP/RTP protocols. RTSP runs on top of TCP, while RTP runs on UDP. Many firewalls are configured to restrict TCP packets by port number, and are very restrictive on UDP. There are three options for streaming through Firewalls with the streaming server. These options are not mutually exclusive. Typically one or more are used to provide the most flexible setup. The three configurations outlined below are for clients behind a Firewall.

1. Stream via Port 80. This option enables the streaming server to encapsulate all RTSP and RTP traffic inside TCP port 80 packets. Because this is the default port used for HTTP-based web traffic, it will get through most firewalls. However, encapsulating the streaming traffic will lower performance on the network, and require faster client connections to maintain streams. It also increases load on the server.

2. Open the appropriate ports on the Firewall. This allows the streaming server to be accessed via RTSP/RTP on the default ports and provides better use of network resources, lower speeds for client connections and less load on the server. The Ports that need to be open for unrestricted streaming include:

   - TCP Port 80: Used for signaling and streaming RTSP/HTTP (if enabled on server)
   - TCP Port 554: Used for RTSP
   - UDP Ports 6970 - 9999: used for UDP streaming.

     Note: a smaller range of UDP ports can usually be used (typically 6970-6999).
   - TCP Port 7070: Optionally used for RTSP (this port is used by Real Server, and QuickTime/Darwin can also be configured to use this port)

3. Set up a Streaming Proxy Server. The Proxy server is placed in the network DMZ - an area on the network that is in between an external firewall to the Internet, and an internal firewall between the DMZ and the internal network. Using firewall rules, packets with the ports defined above are allowed from the Proxy server to clients through the internal firewall, and also between the proxy server and the Internet via the external firewall. However, clients are not allowed to make direct connections to external resource over those ports. This approach insures that all packets bound for the internal network come through the proxy server, providing an additional layer of network security.

3.5.8.2 Running Streaming Server behind Firewalls

Publicly accessible streaming servers can be placed behind firewalls. However, the ports outlined in 2. above should be opened so that clients have open access to the server over both HTTP and RTSP/RTP. Alternatively, you can run behind a restrictive firewall if you specify port 80 in references to your stream. For example, if the server stream.example.com was placed behind a restrictive firewall, and we wanted to access a movie named "sample.mov" we could use the URL:

```
rtsp://stream.example.com:80/sample.mov
```

The following table summarizes ports used by the streaming server. The arrows indicate the packet flow between client and server. This is for QTSS but the ports used are the same.

Usage	Ports	Protocols	Notes
Responding to messages from clients (such as Play and Pause)	TCP (client initiates → QTSS) 554,7070, 8000, 8001, 80)	RTSP, RTP, RTCP, MP3	Main port is 554. 80 is supported in the QT client as an alternative TCP port. These ports also send data to clients.
Sending media and receiving client status	• UDP data (QTSS → client): 6970–6999, even numbers • UDP status (QTSS ↔ client): 6971–6999, odd numbers • TCP data & status (QTSS ↔ client): 554, 7070, 8000, 8001, 80	RTP RTCP RTSP, RTP,RTCP	Status is required to maintain a connection; if blocked, the server disconnects the client. Same ports used to respond to messages.
Receiving broadcasts	• UDP data (broadcaster → QTSS): 6972–65535, even numbers • UDP RTCP status (broadcaster ↔ QTSS): 6973–65535, odd numbers • TCP (broadcaster initiates → QTSS): 554, 7070, 8000, 8001, 80	RTP RTCP RTSP, RTP, RTCP	• Ports depend on the broadcaster configuration. • Status is required to maintain a connection; if blocked, the server disconnects the broadcaster. • Broadcasters can broadcast over their TCP message connection with the server instead of using UDP ports.
Streaming through server	TCP (client initiates → QTSS): 554, 7070, 8000, 8001, 80	RTSP, RTP, RTCP, MP3	Same ports used to respond to messages and receive TCP broadcasts.
MP3 broadcasts (typical default)	TCP (client → QTSS): 8000	HTTP (Icecast)	
Managing QTSS remotely with Server Admin	TCP (admin client initiates → server): 311		
Managing QTSS remotely with Web Admin	TCP (web browser client initiates → server): 1220	HTTP	

Figure 3.10: DSS ports

3.5.9 Bandwidth Considerations

If all of the expected connections for your QuickTime Streaming Server are the same size, then computing your needed bandwidth is relatively simple. Multiply the maximum number of users you expect to support by the bit rate of the files they will view. Here are two examples:

Thirty clients viewing 1.5 megabits per second (Mbps) files would take up a 45Mbps connection which could be carried on 100Mbps Ethernet LAN.

One thousand clients viewing 22 kilobits per second files (Kbps) would take up a 22Mbps connection which could be carried on 100Mbps Ethernet LAN or externally on a fractional T3.

If the bit rates differ between the files (for example, you serve both over the Internet with 22Kbps files and on a LAN with 1.5Mbps files) your connection is more difficult to compute. You may use the formula above but keep in mind that there is no check on the client's bandwidth use. If you allow five hundred users, there is no method to require four hundred of them to connect to the smaller files. The total bandwidth of the server can be set so that as soon as that bandwidth is reached, new users will be turned away until current users disconnect from the server.

Consider that the server may take the entire bandwidth up to the maximum set in the Admin. Also, consider that other network traffic is not part of this calculation. If there are other services on the network, take care to plan for a network and Streaming Server

configuration that will allow all the expected traffic.

It's generally not a good idea to connect a streaming server to the Internet or local area network by Digital Subscriber Line (DSL) or cable modem. The server will be severely limited by the relatively small bandwidth of DSL and cable modems for uploading data. In some cases, running a server on a DSL connection may break a DSL service agreement. Consult your DSL or cable modem service provider before setting up the server.

When authoring Real-Time Transport Protocol (RTP) streams, make sure they do not exceed 75 percent of anticipated client throughput. For example, don't use a rate higher than 20 kilobits per second (Kbps) for a 28 Kbps modem connection. For a typical 56K modem connection, don't use a rate higher than 31 Kbps. For a T1 (1500 Kbps) client connection, don't use a rate higher than 1125 Kbps.

So Audio is possible over the Internet without problems but video will only run properly in Ethernet LAN preferably above or equal 100Mbps if you are considering streaming HD video.

3.5.10 OS issues

A problem playing video streams with the server running in windows was that sometimes you only get sound with no image. This is probably due to a problem with QuickTime Player mp4 decoder codec but nevertheless the way to solve it is to open a another URL previously to the one you want to see. This is hardly a subtle solution if you want to have an automated video on demand solution with remote controlling possibilities so the best is to avoid windows to deploy the server. Use Linux instead.

3.5.11 Conclusion

Even though real media, windows media and now macromedia flash streaming servers are much more used over the internet and may in fact perhaps be better than Darwin, the mp4 and mp3 streaming ability all in the same server and the 100% compatibility with QuickTime Player and therefore Second Life made it the right choice.

Overall it runs quite well and I even got it to run in Windows Vista after some attempts and here I found something that might become a problem if you run lots of mp3 or media playlists. The server launches a new instance of "MP3Broadcaster.exe" for each mp3 playlist being streamed and uses about 1.5MB of system memory for each which, if you have low system resources, can become a problem.

I couldn't found any information regarding this subject so maybe this just happens on windows. What I could found tells that only one thread is used for streaming so either they say this because they count each of the "MP3Broadcaster" as a process(but since every process has at least one thread their information would be wrong) or this only happens on Windows.

Because Darwin is open source the windows version may in fact run differently from Unix versions and the information I've read, regarding the threads involved, might be only referring to QTSS and I got that wrong.

For more detailed information see:
http://ubuntuforums.org/showthread.php?t=651556
http://soundscreen.com/streaming/firewall.html
http://developer.apple.com/opensource/server/streaming/qtss_admin_guide.pdf

3.6 Phidget RFID

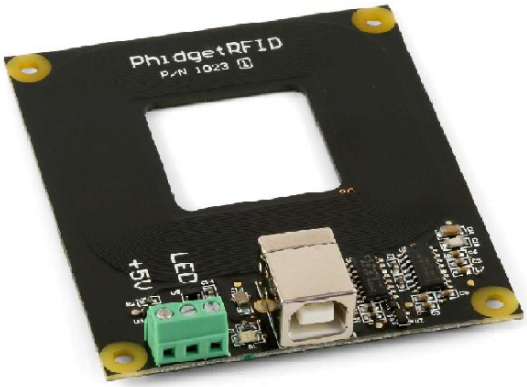

Figure 3.11: PhidgetRFID

3.6.1 Introduction

For "Real World" Sensing we chosen Phidget RFID a easy to use low cost sensing controllable from a PC. Using USB they are managed by an easy to use API. Applications can be developed quickly in C#, Visual Basic, VBA (Microsoft® Access and Excel), LabView, Java, Delphi, C and C++. Announced product features for Phidget RFID are:

- Reads tags brought within 3 inches of the reader
- Reads any tag with EM4102 protocol
- Returns the unique number contained in the tag
- Provides 2 Digital Outputs to drive LEDs, relays, etc.
- On-Board LED
- Connects directly to a computer's USB port

It runs in all Operating Systems(OS).

3.6.2 Setup

Setting up the Phidget RFID is very simple. Just download the drivers for the OS you are using, install them, plug the RFID to a USB port and it should be working. You can test it with the application that comes with the drivers. It only detects tags if the antenna is on so make shore it is.

3.6.3 Tags

The package came with several tags only two were used. The tags are passive and have a unique number which is read by the RFID.

3.6.4 Library

In case you are developing in Java you just need to import the jar with the Phidgets library, which is provided at their web page, read and understand the instructions you need from the API and apply them. It's that simple.

3.6.5 Conclusion

Of course being so simple to work and install you could not expect much but all the basic functions run with no problem at all. You can set it up and running at first try and very quickly, just use one of the examples on the web page and change it to achieve your desired goals.

Only when you start to complicate the running scenarios it starts to fail. If you try to detect more than one tag at the same time it just won't work and it will detect none. This is on the user manual so no complains. Also the tags need to be on top of the detector and I mean that literately. The distance stated in the features is 3 inches (about 7.6 cm) but for my experience I would say it's maybe less.

Overall its simplicity and ease of use make it a good choice to "play". It reminds me of Lego™ (the Technic or Mindstorms kind) and I think that's exactly the concept behind Phidgets.

For more visit: http://www.phidgets.com/

3.7 Virtual World client

Second Life® is an Internet-based virtual world launched in 2003, developed by Linden Research, Inc. Using Avatars as their representation in the Virtual World Residents can explore, meet other Residents, socialize, participate in individual and group activities, create and trade items (virtual property) and services from one another.

3.7.1 Second Life and Firewalls

According to Second Life Support Center these are the ports you need to open and to which Second Life needs to connect to :

- 443/TCP,
- 12035/UDP,
- 12036/UDP,
- 12043/TCP,
- 13000-13050/UDP.

https://support.secondlife.com/ics/support/default.asp?deptID=4417&task=knowledge&questionID=4355

3.8 Conclusion

Streaming and broadcasting content revealed to be much more difficult than I initially thought mostly because of the specifics of the project. The restrictions imposed by the Second Life Client made it impossible to use just VLC player as I initially intended. Because the client uses QuickTime Player plugins to play the audio/video content it must be given a valid URL for both audio and video or it won't work. As for the rest of technologies, apart of XML-RPC, most of those expected to be used were and everything went according to plan.

Chapter 4

Requirements and Architecture

4.1 Introduction

This chapter addresses the architecture and makes an overview of the system's possibilities and interactions to give a more abstract (generic or high level) perspective of the system before addressing the system's implementation details in the next chapter.

4.2 Requirements

System requirements can be divided according to two main points of views: the user and the developer points of view.

4.2.1 User point of view

The basic objectives of our application from a user point of view are as follows:

- The user will be given the possibility of choosing the media type i.e. music or video;
- The choosing of media will be done trough the interaction with objects representing Media Playing devices in Second Life and with interaction with the real device in Real World;
- User detection and Avatar detection, using sensors;
- Possibility of playing multimedia in virtual device when there is no access to a physical supporting device i.e. Virtual support to play the media content when there is no real support available;
- Media playing in real world and Second Life.

4.2.2 Application developer point of view

From a developer point of view objectives are as follows:

- Know the type of media to be used in order to find the best way to play it;
- Minimize audio/video latency;
- Use of the same programming language, if possible, for easier integration;
- Applications must speak the same language i.e. use the same communication protocol;
- Create or apply a communication protocol;
- Develop a support for media playing in Virtual world;
- Develop a Second Life Client to handle communication.

These were the requirements defined in the proposal of this work and even though some of them where not explicitly developed they are implicit in some of the Use Cases that will be address next.

4.3 Use Case

In this section the requirements will be modeled with UML use case diagrams for better understanding of the agents involved and the kind of interactions they can do with the system.

4.3.1 System's Use Case diagram

Even though this is a Mixed Reality application and the most of the use cases take place in both worlds they were divided by Virtual World and Real World for better organizing and understanding but to emphasize the interactions of the Virtual World in the Real World and vice versa Second Life and iPlay, which is the application running in the Real World, where considered to be agents.

And in fact they are since they interact with the system but they could also be considered part of it. The same thing with the Phidget RFID. I considered it to be an agent because it is himself an independent system and it also interacts with the system. The User is the person with a tag in the real world and the Avatar is of course the user representation in the Virtual World with a virtual tag.

The actual person can be at the same time in both Worlds and if the tag's it is using are the same, virtual tag 1 in the Virtual World and tag 1, for the system it will be the same person.

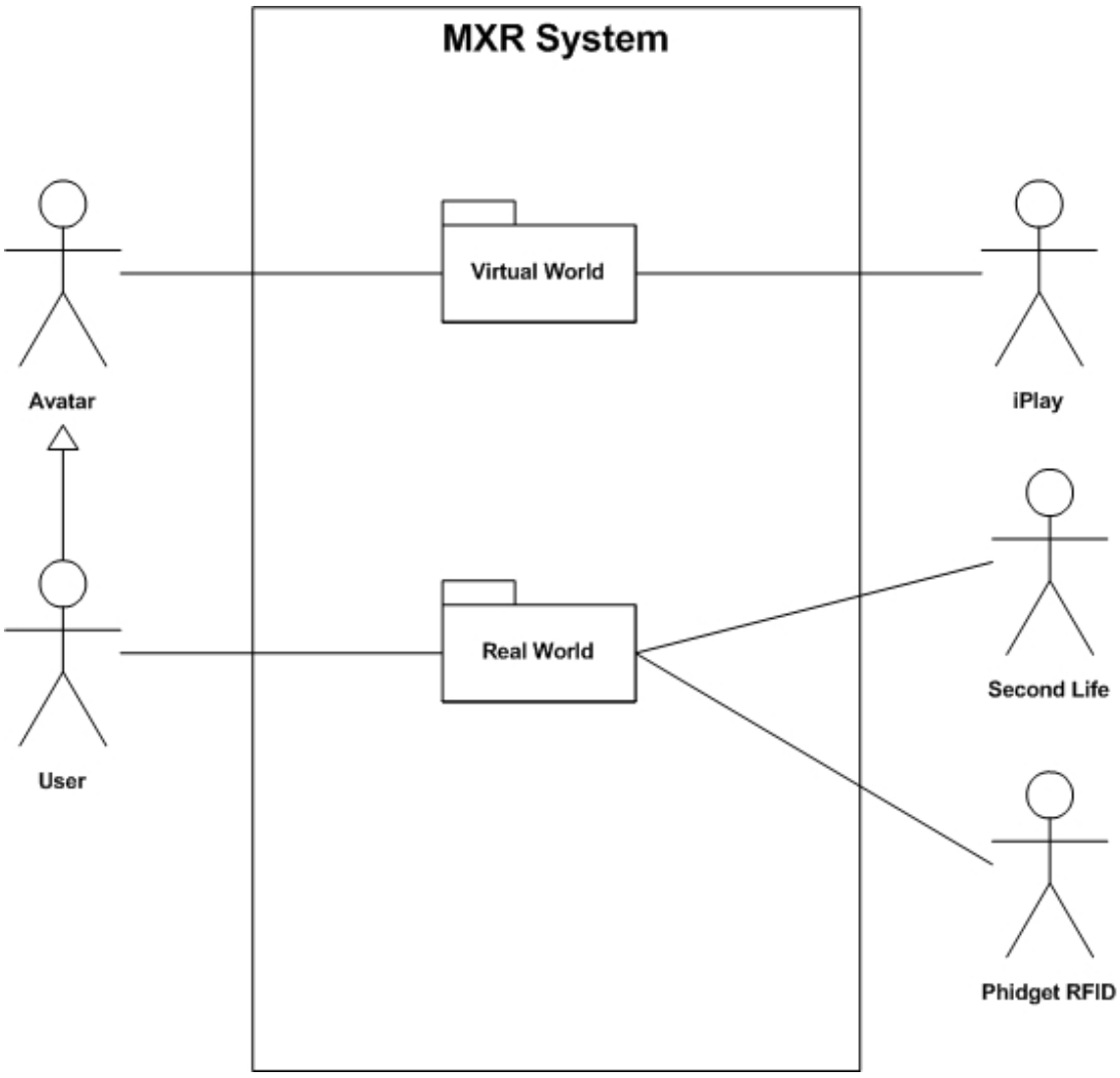

Figure 4.1: Use Case System diagram

4.3.2 Real World diagrams

The Real World package was divided in three major Packets Play, Auto Play and Playlist management which is considered an extend because it is an extra feature, an option. Next each of them will be explained in detail.

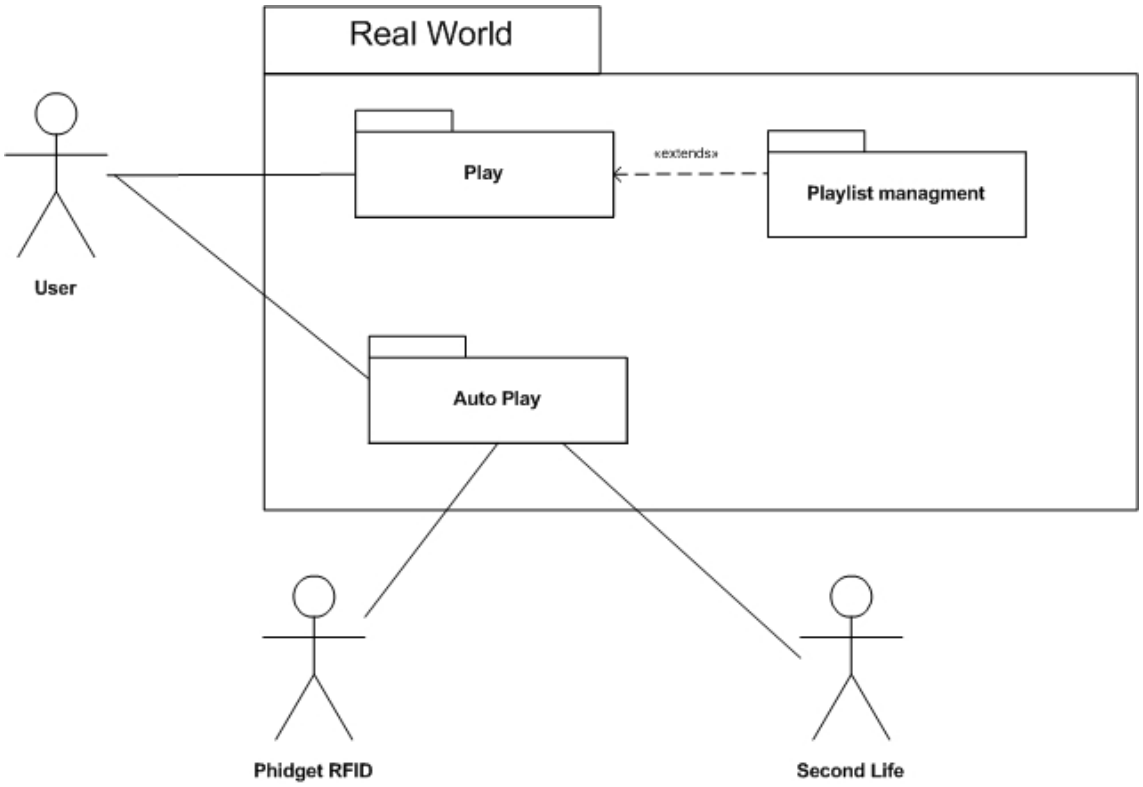

Figure 4.2: Use Case Real World Package diagram

4.3.2.1 Auto Play Package diagram

Now each of the use cases in the package will be explained.

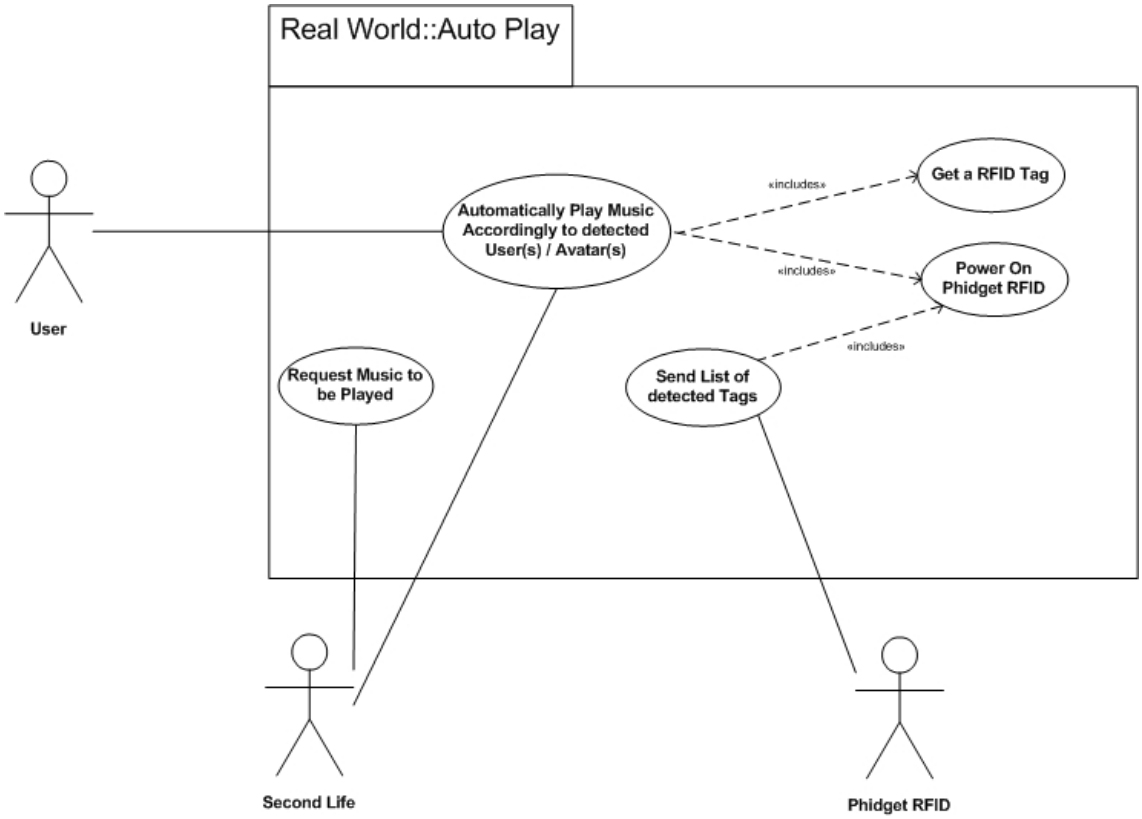

Figure 4.3: Use Case Real World Auto Play Package diagram

Get a RFID Tag

- Description: This use case just consists in the User getting a Tag so it can be detected by the system.
- Sequence of events: The User picks up a Tag.
- Prerequisites: none.
- Agents involved: none.
- Components involved: none.
- Classes involved: none.

(Components Diagram 4.5.2.1)
(Conceptual class diagram 4.4)

Power on Phidget RFID

- Description: This use case consists in the User connecting the Phidget RFID.
- Sequence of events: User connects the Phidget RFID by clicking the RFID button on the iPlay interface.
- Prerequisites: none.
- Agents involved: User.
- Components involved: iPlay.jar,phidgets.jar.
- Classes involved: IPlay,Phidget RFID.

(Components Diagram 4.5.2.1)
(Conceptual class diagram 4.4)

Send List of detected Tags

- Description: The Phidget RFID sends information about the detected tag to the main application.
- Sequence of events: User approaches the Tag to the Phidget RFID, the Phidget RFID detects Tag and sends this information to the main application.
- Prerequisites: Phidget RFID is ON and User approached Tag.
- Agents involved: Phidget RFID, User.
- Components involved: iPlay.jar,phidgets.jar.
- Classes involved: IPlay,Phidget RFID.

(Components Diagram 4.5.2.1)
(Conceptual class diagram 4.4)

Request Music to be played

- Description: The main application receives a request from Second Life to play a specific music. This request came from an Avatar in Second Life but from the Systems Real World point of view it's as if Second Life is an agent that interacts with the system by requesting for a song to be played.
- Sequence of events: Second Life requests song and application launches a player that plays it.
- Prerequisites: none.

- Agents involved: Second Life.
- Components involved: Second Life(Client), iPlay.jar, HTTP.jar, Second Life HTTP Server, MP3Broadcaster, and Firefox.
- Classes involved: Music, Audio Playlist, IPlay, and SLServer.

(Components Diagram 4.5.2.1)
(Conceptual class diagram 4.4)

Automatically Play Music accordingly to detected User(s)/Avatar(s)

- Description: User detection in Real World triggers a chain of events that will reflect in both worlds. If it is an Avatar detected from the Real World point of View it's the same thing as if the Agent Second Life asks the System to play a specific music.
- Sequence of events: User is detected, information is sent to Virtual World, application in Virtual World decides the new Mixed-Reality state, Virtual World Application sends back information to Real World with the new state and finally the same music is played in both worlds according to the final state.
- Prerequisites: Phidget RFID sent info on detected Tag.
- Agents involved: User, Second Life.
- Components involved: all (except video related ones).
- Classes involved: all (except video related ones).

(Components Diagram 4.5.2.1)
(Conceptual class diagram 4.4)

Being a bit more complex than the other use cases in the package an Activity Diagram was added to this use case for better understanding of how it works. This involves details from implementation and the actual functioning of the system so it's better to read the next chapter first to better understand it.

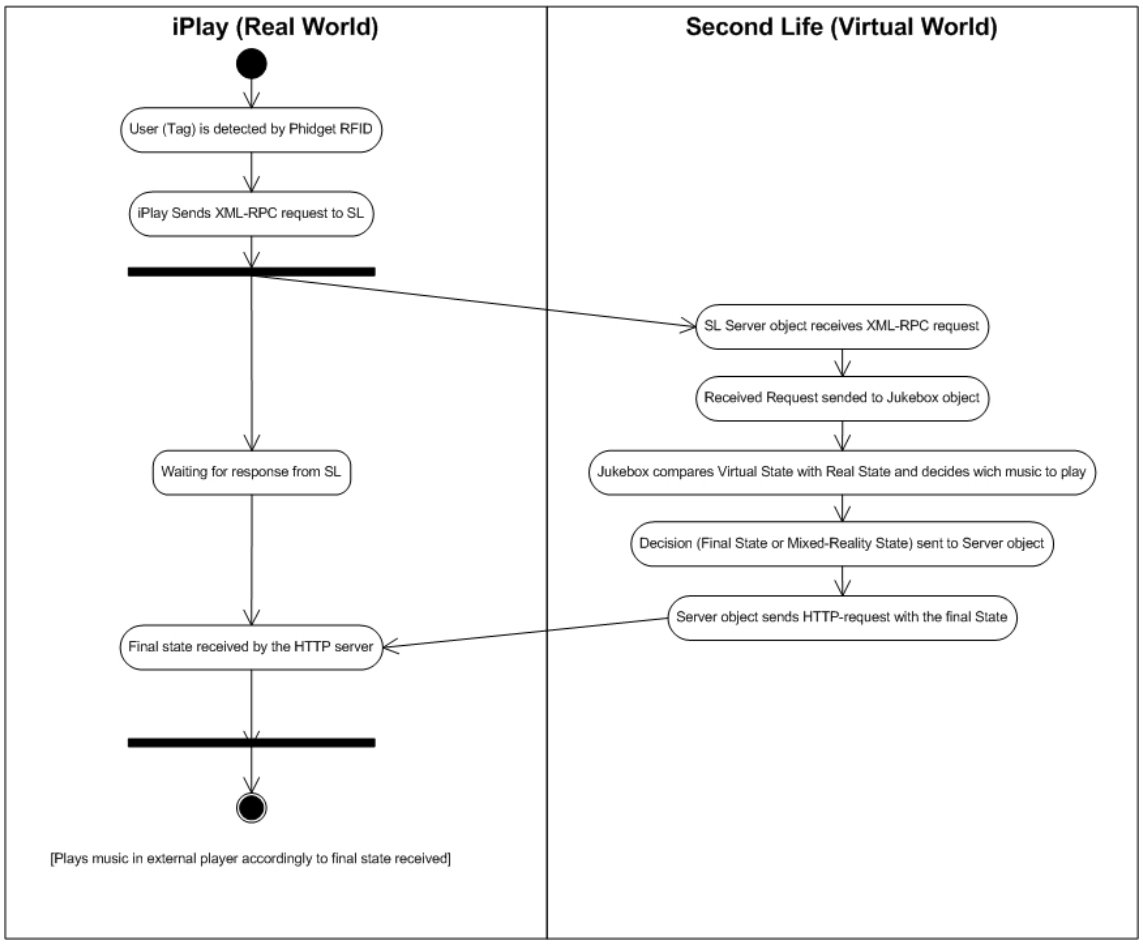

Figure 4.4: Activity Real World Auto Play diagram

4.3.2.2 Play Package diagram

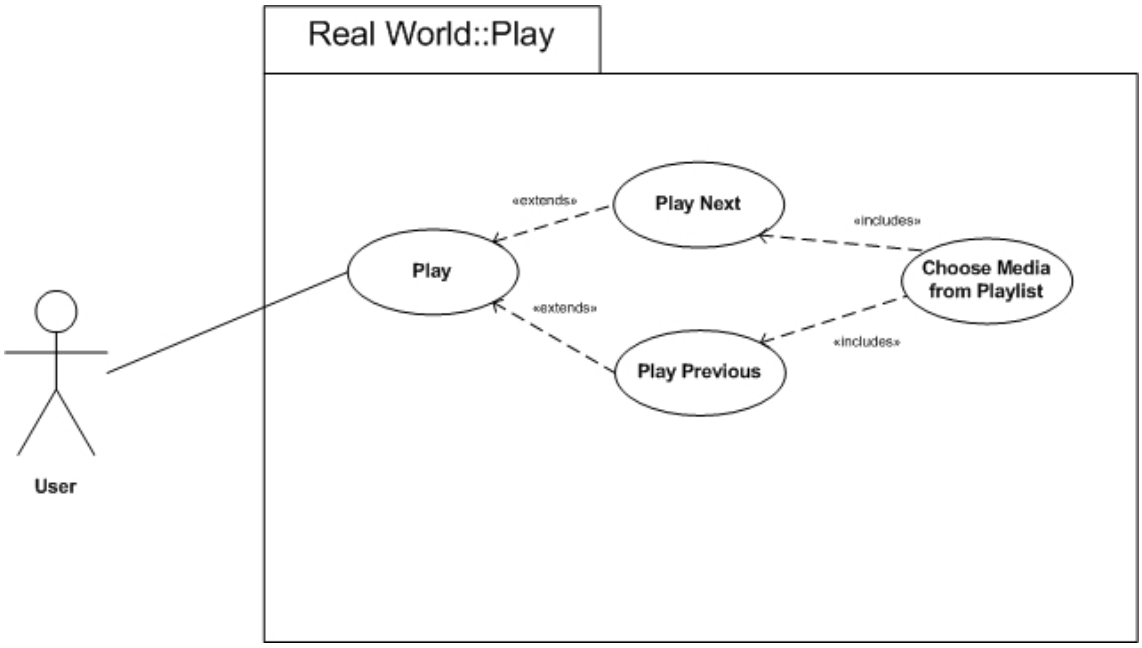

Figure 4.5: Use Case Real World Play Package diagram

Play

- Description: User plays a music.
- Sequence of events: User press Play button on the interface and the selected music will play. If the button SL is on pressing the Play button sends the music information to Virtual World and it will play in both worlds.
- Prerequisites: none.
- Agents involved: User.
- Components involved: iPlay.
- Classes involved: IPlay, Music.

(Components Diagram 4.5.2.1)
(Conceptual class diagram 4.4)

Play Next

- Description: User plays the next music in Playlist. This is an extension (an option) from normal play and it includes the Choose media from Playlist Use Case because by pressing the Fast Forward (FF) button you are choosing a different song from the playlist.

- Sequence of events: User press FF Button and then the Play button on the interface and the selected music will play. If the button SL is on pressing the Play button sends the music information to Virtual World and it will play in both worlds.
- Prerequisites: none.
- Agents involved: User.
- Components involved: iPlay.
- Classes involved: IPlay, Music, and Audio Playlist.

(Components Diagram 4.5.2.1)
(Conceptual class diagram 4.4)

Play Previous

- Description: User plays the previous music in Playlist. This is an extension (an option) from normal play and it includes the Choose media from Playlist Use Case because by pressing the Fast Backward (FB) button you are choosing a different song from the playlist.
- Sequence of events: User press FB Button and then the Play button on the interface and the selected music will play. If the button SL is on pressing the Play button sends the music information to Virtual World and it will play in both worlds.
- Prerequisites: none.
- Agents involved: User.
- Components involved: iPlay.
- Classes involved: IPlay, Music, and Audio Playlist.

(Components Diagram 4.5.2.1)
(Conceptual class diagram 4.4)

Choose media from Playlist

- Description: User chooses a different music in Playlist.
- Sequence of events: User press FB or FF Button on the interface and the selected music will change.
- Prerequisites: none.
- Agents involved: User.

- Components involved: iPlay.
- Classes involved: IPlay, Music, and Audio Playlist.

(Components Diagram 4.5.2.1)
(Conceptual class diagram 4.4)

4.3.2.3 Playlist Management Package diagram

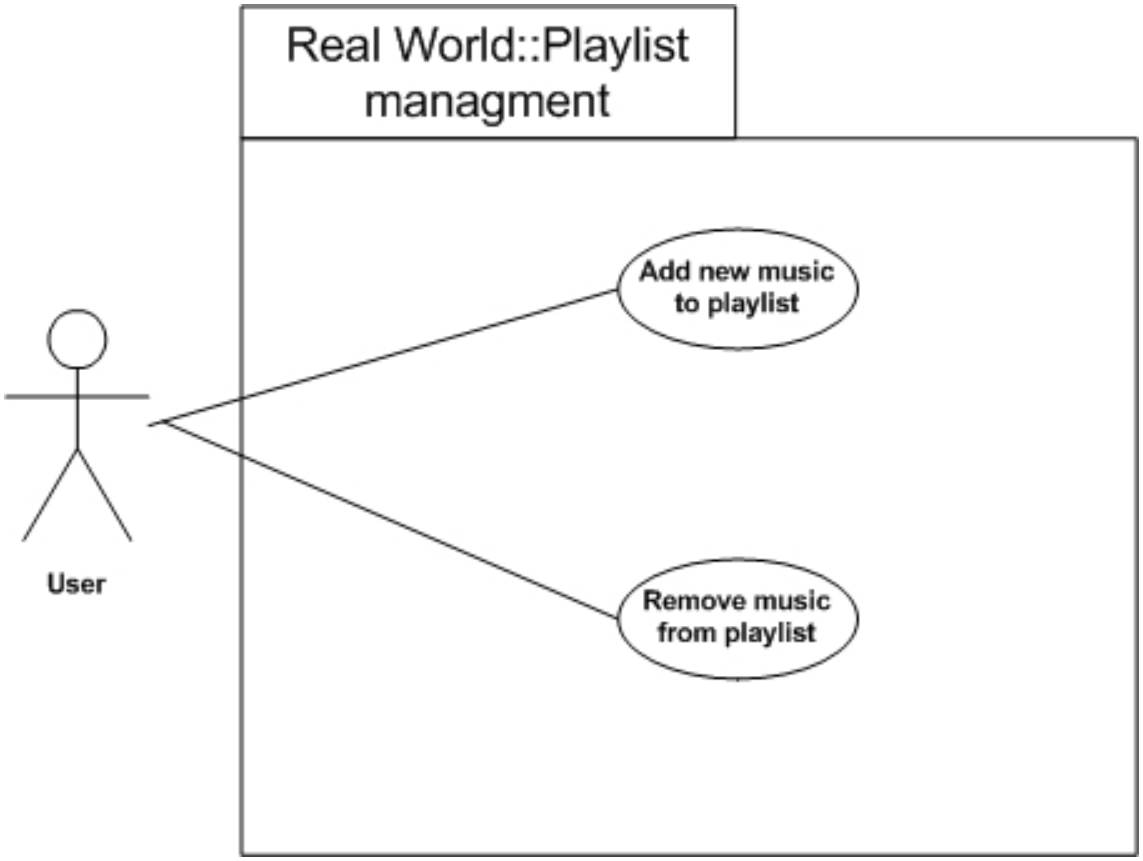

Figure 4.6: Use Case Real World Playlist Management Package diagram

Add new music to playlist

- Description: User adds a new music to playlist by editing the file. (A specific interface was developed for this but is not active)
- Sequence of events: User opens the text file and adds a new line with the music.
- Prerequisites: none.
- Agents involved: User.
- Components involved: Playlist.txt.
- Classes involved: none.

(Components Diagram 4.5.2.1)
(Conceptual class diagram 4.4)

Remove music from playlist

- Description: User removes music from playlist by editing the file.
- Sequence of events: User opens the text file and deletes the line with the music.
- Prerequisites: none.
- Agents involved: User.
- Components involved: Playlist.txt.
- Classes involved: none.

(Components Diagram 4.5.2.1)
(Conceptual class diagram 4.4)

4.3.3 Virtual World diagrams

The Virtual World package was divided in three major Packets (just like in the Real World) Play, Auto Play and Playlist management which is considered as extend because it is an extra feature, an option. Next each of them will be explained in detail.

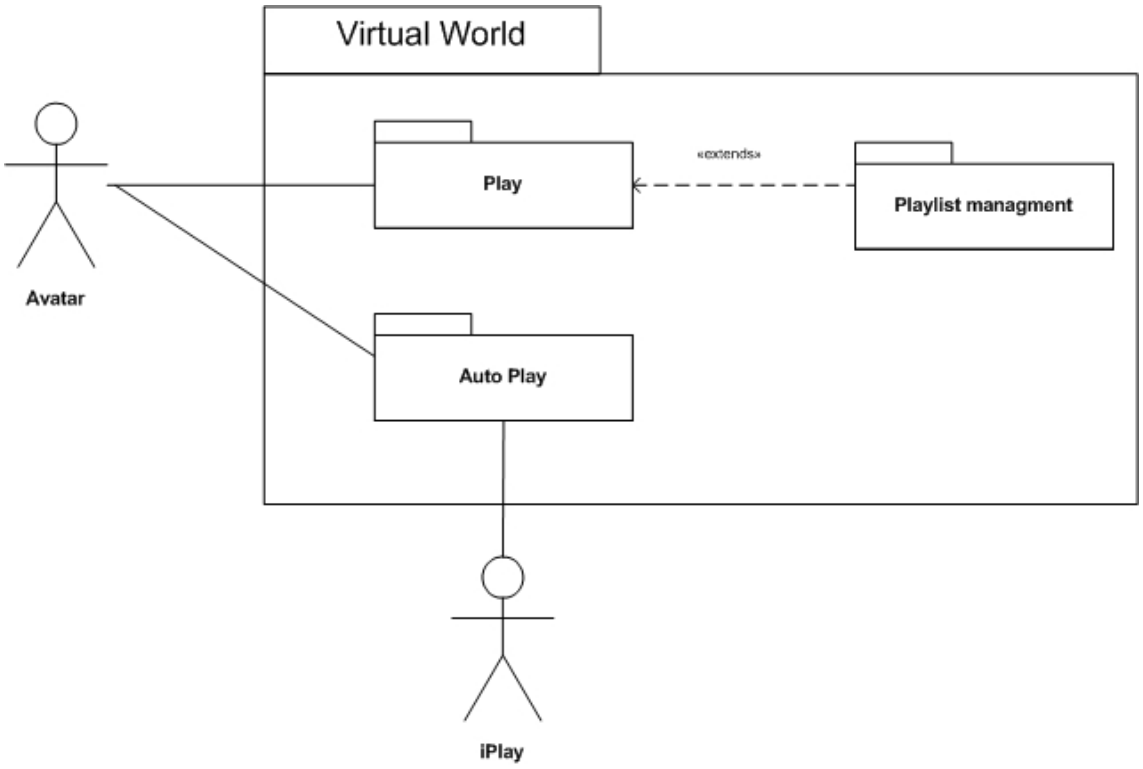

Figure 4.7: Use Case Virtual World Package diagram

4.3.3.1 Auto Play Package diagram

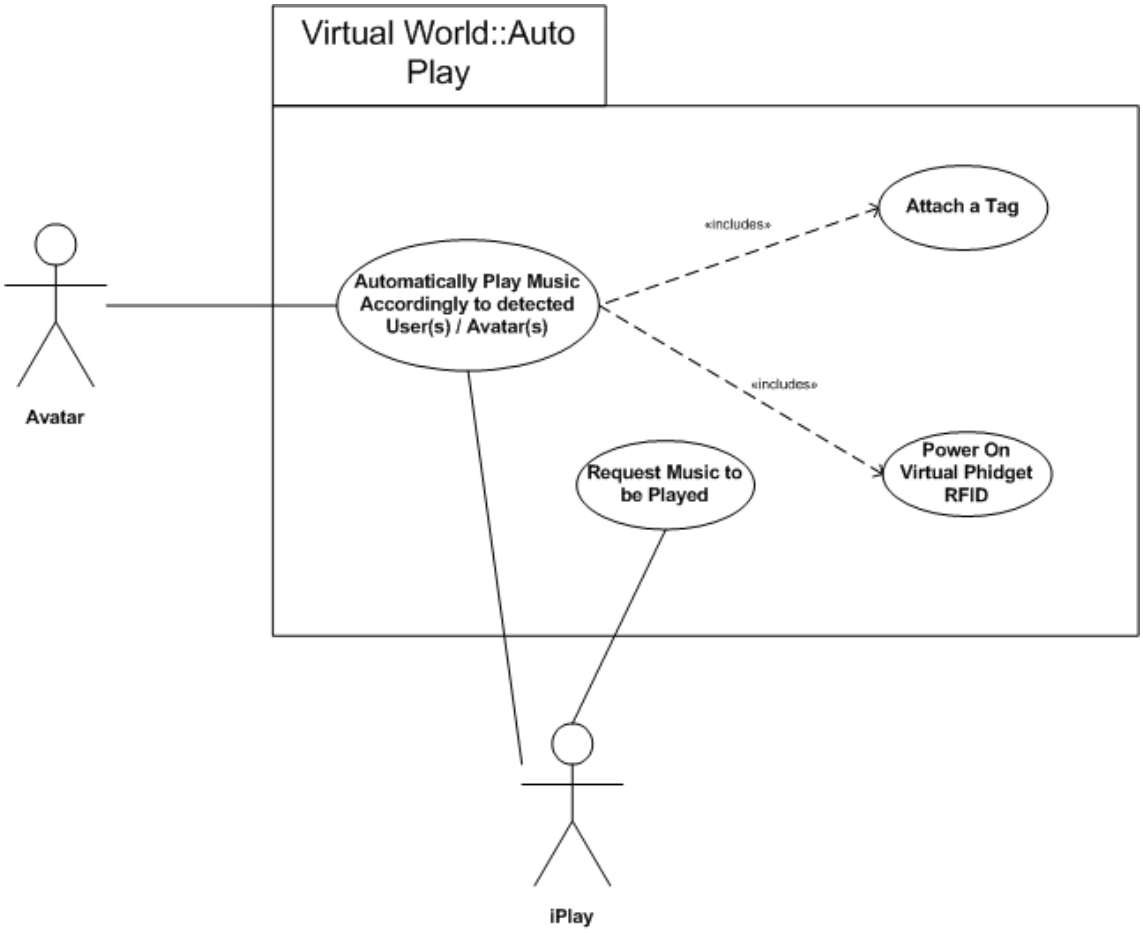

Figure 4.8: Use Case Virtual World Auto Play Package diagram

Attach a Tag

- Description: This Use Case consists in Attaching a Tag so the Avatar can be detected by the system.
- Sequence of events: The Avatar touches a Tag and accepts the attachment request.
- Prerequisites: none.
- Agents involved: Avatar (or can be considered none because until the Avatar has a Tag attached it does not exist to the System).
- Components involved: Second Life(Client).
- Classes involved: VirtualTag.

(Components Diagram 4.5.2.1)
(Conceptual class diagram 4.4)

Power on Virtual Phidget RFID

- Description: This use case consists in the Avatar starting the Virtual Phidget RFID.
- Sequence of events: User connects the Virtual RFID by clicking the object.
- Prerequisites: none.
- Agents involved: Avatar.
- Components involved: Second Life(Client).
- Classes involved: VirtualRFID.

(Components Diagram 4.5.2.1)
(Conceptual class diagram 4.4)

Request Music to be played

- Description: The main application receives a request from iPlay to play a specific music. This request came from a User in Real World but from the Systems Virtual World point of view it's as if iPlay is an agent that interacts with the system by requesting for a song to be played.
- Sequence of events: iPlay requests song and application plays it.
- Prerequisites: none.
- Agents involved: iPlay.
- Components involved: Second Life(Client), iPlay.jar, XML-RPC.jar, Second Life XML-RPC Server, and MP3Broadcaster.
- Classes involved: Music, Audio Playlist, Jukebox, and Server.

(Components Diagram 4.5.2.1)
(Conceptual class diagram 4.4)

Automatically Play Music accordingly to detected User(s)/Avatar(s)

- Description: An Avatar detection in Virtual World triggers a chain of events that will reflect in both worlds. If it is a User detected from the Virtual World point of View it's the same thing as if the Agent iPlay asks the System to play a specific music.
- Sequence of events: Avatar is detected, information is sent to main application, application decides the new Mixed-Reality state, application sends information to Real World with the new state and finally the same music is played in both worlds according to the final state.

- Prerequisites: Virtual Phidget RFID is on and detecting and the Avatar have a Tag attached.
- Agents involved: Avatar, iPlay.
- Components involved: all (except video related ones).
- Classes involved: all (except video related ones).

(Components Diagram 4.5.2.1)
(Conceptual class diagram 4.4)

Being a bit more complex than the other use cases in the package an Activity Diagram was added to this use case for better understanding of how it works. Once again this involves details from implementation and the actual functioning of the system so it's better to read the next chapter first to better understand it.

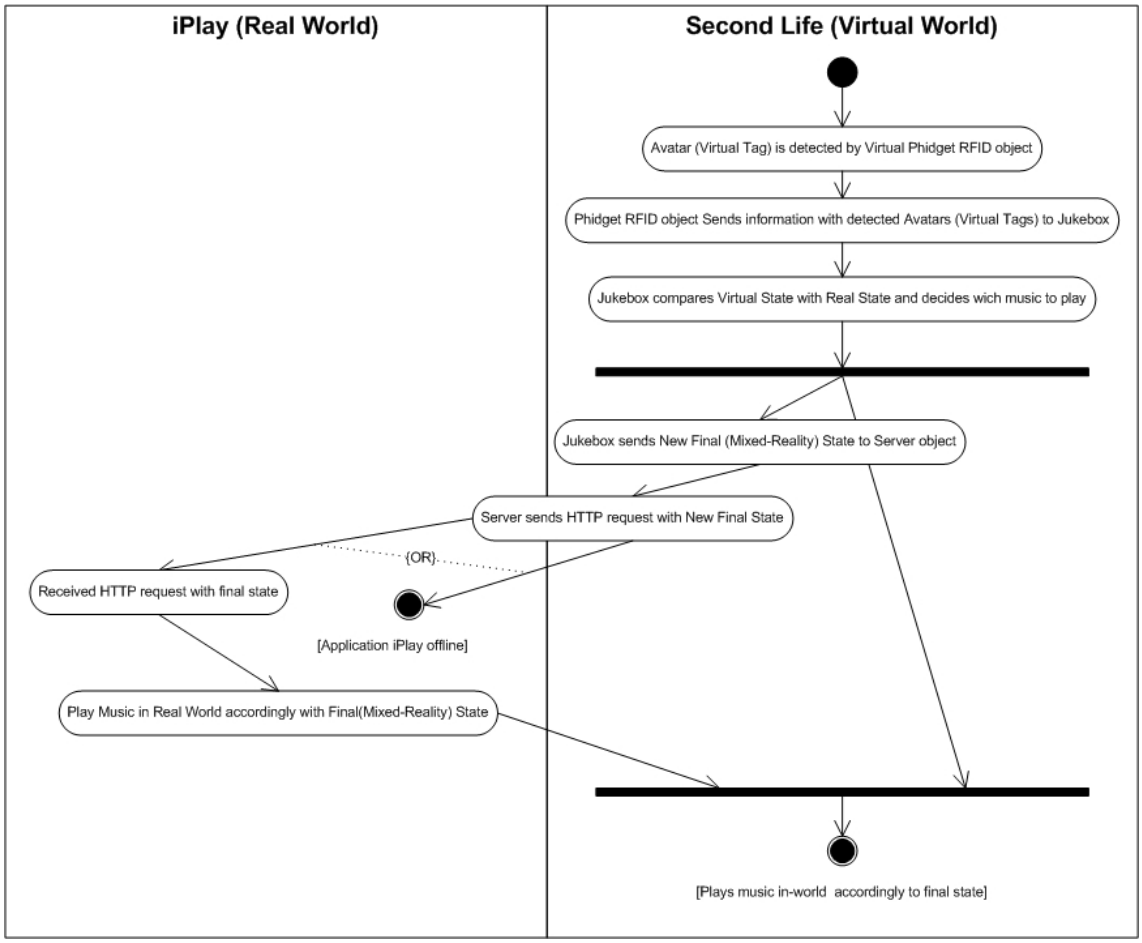

Figure 4.9: Activity Virtual World Auto Play diagram

Activity Virtual World Auto Play diagram

4.3.3.2 Play Package diagram

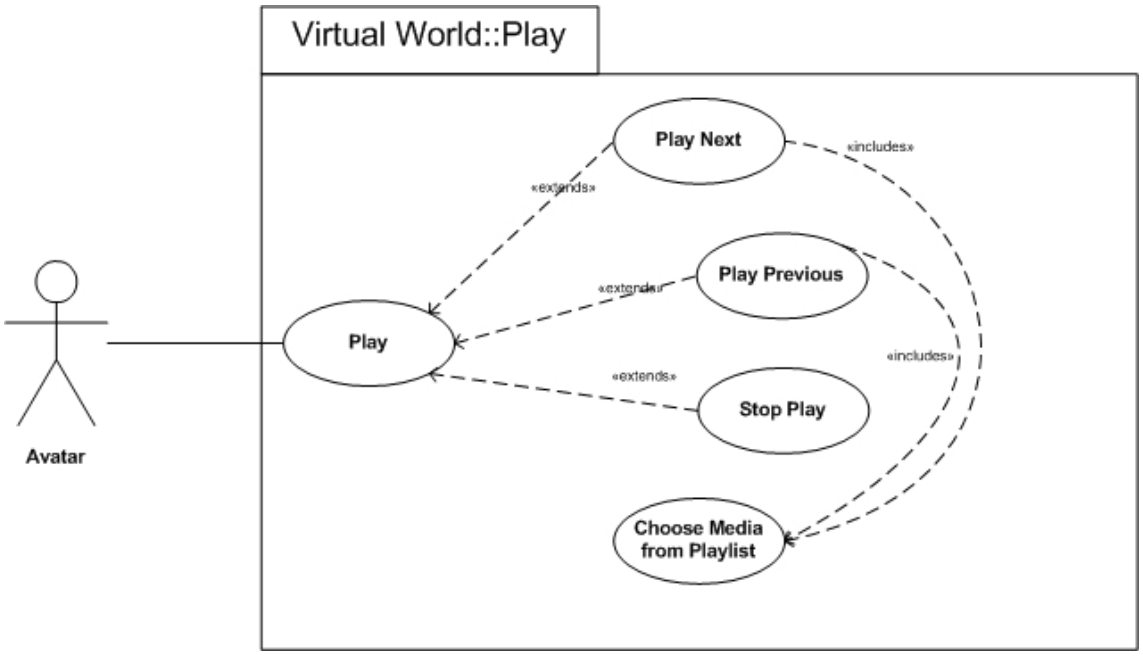

Figure 4.10: Use Case Virtual World Play Package diagram

Play

- Description: Avatar plays music.
- Sequence of events: Avatar clicks on the Jukebox to make the dialog menu appear, then press Play button on the dialog menu from the Jukebox and the music will play. If iPlay is running pressing the Play button sends the music information to Virtual World and it will play in both worlds.
- Prerequisites: none.
- Agents involved: User.
- Components involved: Second Life (Client).
- Classes involved: Jukebox, Music.

(Components Diagram 4.5.2.1)
(Conceptual class diagram 4.4)

Play Next

- Description: Avatar plays the next music in Playlist. This is an extension (an option) from normal play.

- Sequence of events: Avatar clicks on the Jukebox to make the dialog menu appear, then press FF Button and then the Play button on the dialog menu and the selected music will play.
- Prerequisites: none.
- Agents involved: Avatar.
- Components involved: iPlay.
- Classes involved: Jukebox, Music, and Audio Playlist.

(Components Diagram 4.5.2.1)
(Conceptual class diagram 4.4)

Play Previous

- Description: Avatar plays the previous music in Playlist. This is an extension (an option) from normal play.
- Sequence of events: Avatar clicks on the Jukebox to make the dialog menu appear, then press FB Button and then the Play button on the dialog menu and the selected music will play.
- Prerequisites: none.
- Agents involved: Avatar.
- Components involved: Second Life (Client).
- Classes involved: Jukebox, Music, and Audio Playlist.

(Components Diagram 4.5.2.1)
(Conceptual class diagram 4.4)

Stop Play

- Description: Avatar stops the playing music. This is an extension (an option) from normal play.
- Sequence of events: Avatar clicks on the Jukebox to make the dialog menu appear, then press Stop Button and then the music will stop playing.
- Prerequisites: none.
- Agents involved: Avatar.
- Components involved: Second Life (Client).

- Classes involved: Jukebox, Music, and Audio Playlist.

(Components Diagram 4.5.2.1)
(Conceptual class diagram 4.4)

Choose media from Playlist

- Description: User chooses a different music in Playlist. It is included Play Previous and Play next Use Cases because by FF or FB you are implicitly choosing a different song from the playlist.
- Sequence of events: User press FB or FF Button on the dialog menu and the selected to play music will change.
- Prerequisites: none.
- Agents involved: Avatar.
- Components involved: Second Life (Client).
- Classes involved: Jukebox, Music, and Audio Playlist.

(Components Diagram 4.5.2.1)
(Conceptual class diagram 4.4)

4.3.3.3 Playlist Management Package diagram

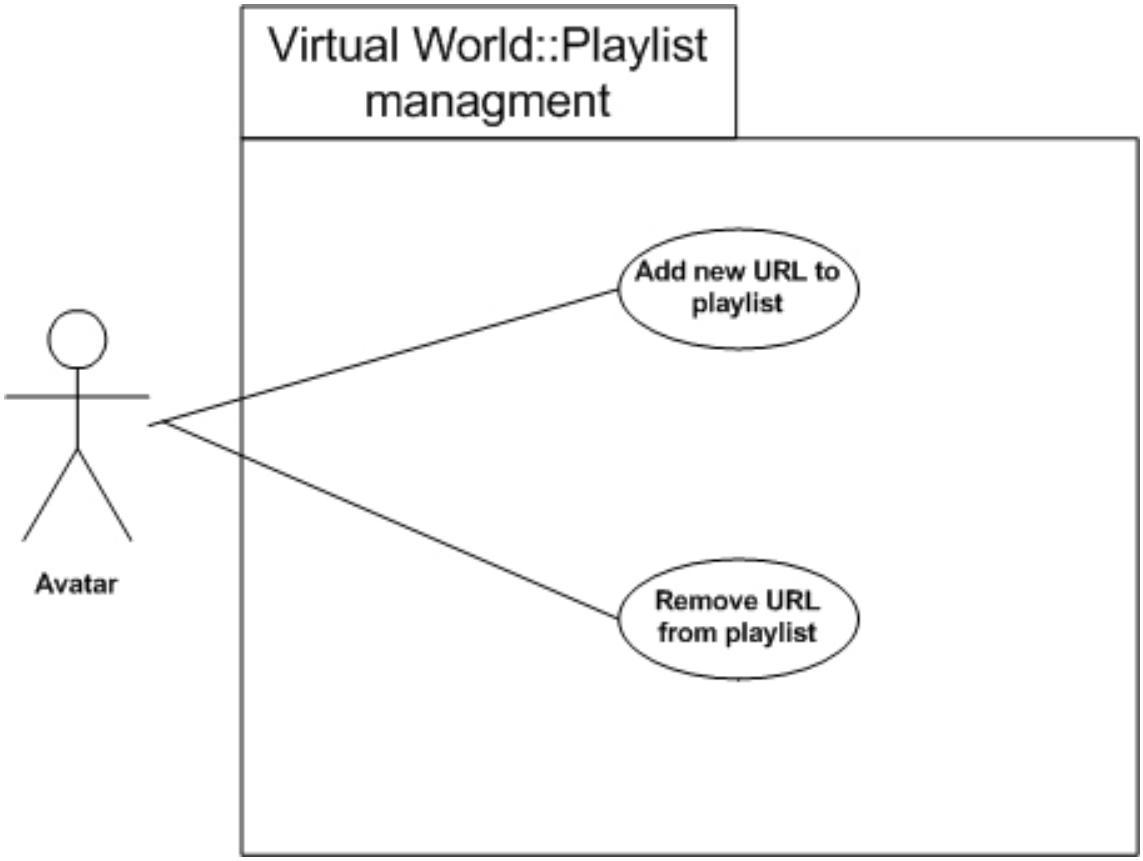

Figure 4.11: Use Case Virtual World Playlist Management Package diagram

Add new URL to playlist

- Description: Avatar adds a new music to playlist by editing the notecard.
- Sequence of events: Avatar clicks Jukebox with right mouse button and chooses to edit the object, then goes to the tab where scripts and notecards are located, opens the notecard file and adds a new line with the music URL.
- Prerequisites: none.
- Agents involved: Avatar.
- Components involved: none.
- Classes involved: none.

(Components Diagram 4.5.2.1)
(Conceptual class diagram 4.4)

Remove URL from playlist

- Description: Avatar removes a music from the playlist by editing the notecard.
- Sequence of events: Avatar clicks Jukebox with right mouse button and chooses to edit the object, then goes to the tab where scripts and notecards are located, opens the notecard file and removes the line which contains the music URL.
- Prerequisites: none.
- Agents involved: Avatar.
- Components involved: none.
- Classes involved: none.

(Components Diagram 4.5.2.1)
(Conceptual class diagram 4.4)

4.4 Domain Class Model

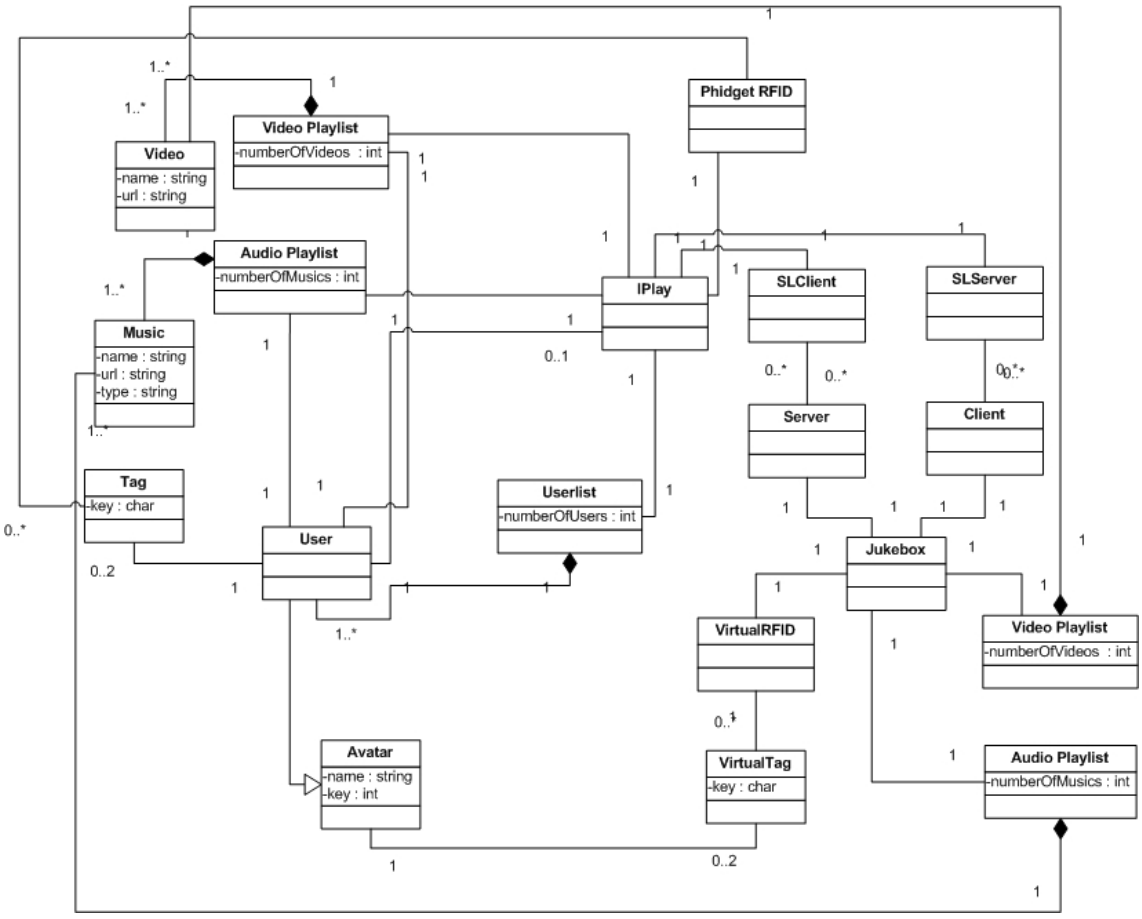

Figure 4.12: Conceptual Class Diagram

With the Conceptual Class Diagram we can see the classes of the system, their interrelationships, and the operations and attributes of the classes. This serves to analyze requirements in the form of a conceptual/analysis model. Next the classes will be described one by one. To make it easier to understand the classes methods where ignored and being this a conceptual model and not deployment actual implementation details will not be addressed. Also keep in mind this is the concept so not all the classes where actually deployed and some of them may not work exactly the way they are described in next.

4.4.1 Classes description

4.4.1.1 IPLay

- Description:IPlay is the main class of the Real World application, it implements the interface and starts all others and itself.
- Relations: 1:1 with SLCliente, SLServer, Phidget RFID, Userlist, Audio Playlist and Video Playlist; 0..1:1 with User;

4.4.1.2 Phidget RFID

- Description: Conceptual representation of the Phidget RFID;
- Relations: 1:1 with IPlay; 1:0..* with Tag;

4.4.1.3 Tag

- Description: Conceptual representation of the RFID Tag;
- Attributes:

  -key: char (the unique key read from the Tag)
- Relations: 0..*:1 with Phidget RFID; 0..2:1 with User (only two tags used);

4.4.1.4 User

- Description: representation of the user, the user only exists to the system if it has a tag with him;
- Relations: a composition aggregation 1..*:1 relation with Userlist meaning the userlist is composed of users and does not make sense without them (does not exist); Inheritance with Avatar meaning it is a specialization of the Avatar class or a child class which may sound weird but in fact the user can do all the things the Avatar does and more because the User can be also an Avatar but the contrary is not true; 1:0..2 with Tag; 1:1 with Audio and Video Playlist and with IPlay;

4.4.1.5 Userlist

- Description: list of the users recognized by the system;
- Attributes:

  -numberOfUsers: int;
- Relations: a composition aggregation 1:1..* relation with User; 1:1 with IPLay;

4.4.1.6 Music

- Description: represents the music stream. Only one class was represented for the playlist in both Worlds to emphasize the fact that the stream is the same. Same thing for the Video class;
- Attributes:

  -name: string

  -URL: string (the URL of the stream with which the music can be accessed)

  -type: string (type of music: soul, jazz...)

- Relations: a composition aggregation 1..*:1 relation with Audio Playlist in both Worlds;

4.4.1.7 Video

- Description: similar to the Music class but foe the video stream;
- Attributes:

  -name: string

  -URL: string (the URL of the stream with which the video can be accessed)

- Relations: a composition aggregation 1..*:1 relation with Video Playlist in both Worlds;

4.4.1.8 Audio Playlist

- Description: represents the audio playlist available. There are two instances of this same class in the diagram because one is in the Real World and the other is in the Virtual what distinguish them is that the Real World one has a relation with IPlay and the Virtual World one with Jukebox;
- Attributes:

  numberOfMusics:int;

- Relations: a composition aggregation 1:1..* relation with Music; 1:1 with iPlay (if Real World one); 1:1 with Jukebox (if Virtual World one);

4.4.1.9 Video Playlist

- Description: similar to Audio Playlist but for Video;
- Attributes:

  numberOfVideos:int;

- Relations: a composition aggregation 1:1..* relation with Video; 1:1 with iPlay (if Real World one); 1:1 with Jukebox (if Virtual World one);

4.4.1.10 SLClient

- Description: XML-RPC Client, sends requests to Virtual World;
- Relations: 1:1 with iPlay; 0..*:0..* with Server;

4.4.1.11 SLServer

- Description: HTTP Server, receives requests from Virtual World;
- Relations: 1:1 with iPlay; 0..*:0..* with Client;

4.4.1.12 Client

- Description: HTTP Client, sends requests to Real World;
- Relations: 1:1 with Jukebox; 0..*:0..* with SLServer;

4.4.1.13 Server

- Description: XML-RPC Server, receives requests from Real World;
- Relations: 1:1 with Jukebox; 0..*:0..* with SLClient;

4.4.1.14 Jukebox

- Description: Jukebox is the main class of the Virtual World application, it is the class that decides the final Mixed-Reality state so it can be considered the main Mixed-Reality class.
- Relations: 1:1 with Client, Server, VirtualRFID, Audio Playlist and Video Playlist;

4.4.1.15 VirtualRFID

- Description: Conceptual Virtual representation of the Phidget RFID;
- Relations: 1:1 with Jukebox; 1:0..* with VirtualTag;

4.4.1.16 VirtualTag

- Description: Conceptual Virtual representation of the RFID Tag;
- Attributes:

  -key: char (the unique key read from the Tag)
- Relations: 0..*:1 with VirtualRFID; 0..2:1 with User (only two VirtualTags used);

4.4.1.17 Avatar

- Description: representation of the Avatar, like the user it only exists to the system if it has a tag attached , in this case, to him;
- Relations: Inheritance with User (see explanation in class User 4.4.1.4); 1:0..2 with VirtualTag;

4.5 Architecture

4.5.1 Logical Architecture

The Logical Architecture gives you an idea about the distribution of the logic constitution of the system, the Classes.

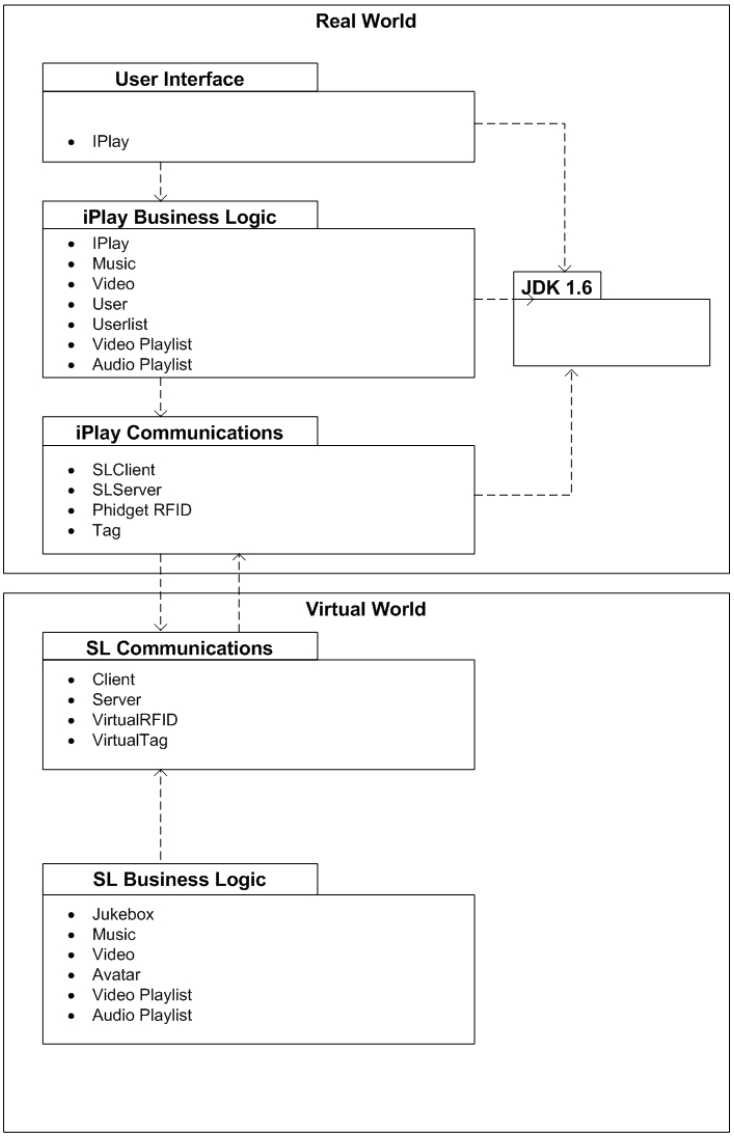

Figure 4.13: Logical Architecture Diagram

Here you can see how the Classes can be classified in terms of which World they are located and the function they perform in the Mixed-Reality System. Notice the Java package in the Real World representing the Java Virtual Machine.

4.5.2 Physical Architecture

4.5.2.1 Components Diagram

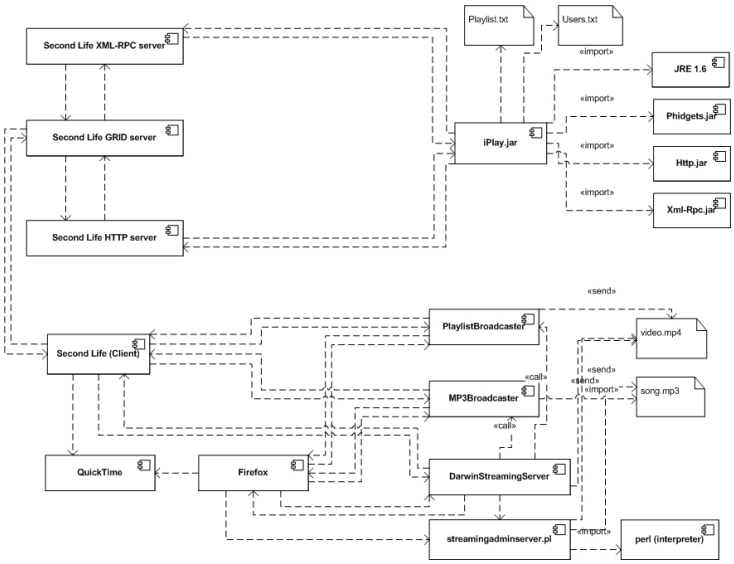

Figure 4.14: Components Diagram

Even though the components in this diagram are most of them coincident with the final implemented product the components diagram focus on identifying the initial architectural landscape for the system so the deployed system may or may not be this way. In the diagram you can see the components and their connections with the rest of the pieces that compose the system. Next a brief description of the components and what they represent:

- iPlay.jar - This is the main application in the Real World, it communicates with its counterpart, the main application in the Virtual World, provides the user interface in Real World, enabling the control of the RFID by the user, plays the multimedia in Real World;

- Playlist.txt - this is a text file with the music's URL, it's accessed and modified by the iPlay application;

- Users.txt - a text file with information to associate a user preferences with a specific tag, it's accessed and modified by the iPlay application;

- JRE 1.6 - representation of the Java Virtual Machine at running time, it's used by all running java applications serving as an additional layer which can be bad turning the running application slower but with the advantage of OS independence.

- Phidgets.jar - jar containing the libraries needed to operate the Phidgets from within a Java application,

- Http.jar - jar containing the HTTP libraries needed to implement a server to receive requests from the virtual world. It is only needed if JRE is lower than 1.6 so in this case it's not really necessary;
- Xml-Rpc.jar - jar containing the XML-RPC libraries needed to implement a client to send requests for the virtual world;
- Second Life XML-RPC server - Linden Labs XML-RPC server;
- Second Life HTTP server - Linden Labs HTTP server;
- Second Life Grid server - Linden Labs Grid server, this is the server providing the Virtual World;
- Second Life (Client) - The client (viewer) of the Virtual World Second Life;
- QuickTime (Plugin) - used by Firefox and Second Life (Client) to play multimedia content;
- Firefox - Firefox browser that can be used to edit playlist from Darwin Streaming Server or to play the contents it provides;
- PlaylistBroadcaster - it's a part of Darwin Streaming Server and it streams the video playlists defined in the admin interface;
- MP3Broadcaster - also part of Darwin Streaming Server it streams the mp3 playlists defined in the admin interface;
- DarwinStreamingServer - the main application of Darwin controlling all others and streaming the video on demand content;
- streamingadminserver.pl - written in Perl it's the admin web interface of the Darwin Streaming Server that enables users to control the server from web browser;
- Perl(interpreter) - used by streamingadminserver.pl;
- video.mp4 - represents a video file to be streamed by Darwin;
- song.mp3 - represents an audio file to be streamed by Darwin;

4.5.2.2 Deployment Diagram

Deployment Diagram models the hardware used in implementing a system, the components deployed on the hardware, and the association between those hardware components.

As you can see in the diagram the Second Life client where considered to be in the same machine but they can be in different machines. The connection between the Darwin Streaming Server and the client machine it's direct it does not connect to Second Life

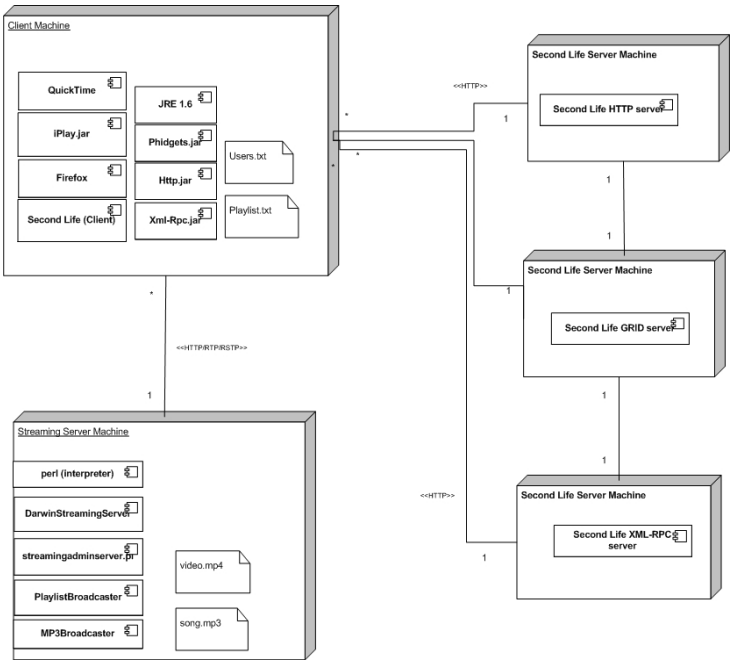

Figure 4.15: Deployment Diagram

servers. The connections are HTTP for receiving the audio stream and connect to the admin web interface of Darwin, RTP for video streaming, and RTSP for video on demand. RTP uses UDP while RTSP uses TCP which makes RTP faster but more vulnerable to firewalls because UDP traffic is more restricted than TCP. The connection of the Second Life client with the Grid Server is not specified because I'm not certain of what protocol they use.

4.6 Conclusion

The requirements stated in this chapter were the initially established ones and even though the Use Cases used to model this requirement might not address them directly they are implicit in some of them. The Class Model is only to give an idea of the classes involved in the project and the connection and interactions between them. The architecture should provide the information and structure necessary to develop the application.

Chapter 5

The Mixed-Reality Application

5.1 Introduction

The System is composed by the Virtual and Real World running applications and is a Mixed-Reality application in the sense that it interacts with both worlds and its actions and reactions can be triggered from within the Real or the Virtual World. The next sections will approach the aspects of its implementation first the Virtual World and them the Real World applications.

Second Life was the Virtual World used in the project. LSL, Linden Scripting Language, it's a programming language with C-like syntax that was used to develop the scripts used in Second Life to implement the Virtual world counterpart of the Real World Objects (the RFID, the Music Player...).

iPlay was the name chosen for the developed application mainly because one of the systems used to test and develop it was the Mac Mini so following the Apple names policy the "app" was named iPlay. This is the support for the system parts running in the Real World including the Phidget RFID. In 5.3 will be approached the aspects of this application interface, Designed Class diagram and implementation details.

5.2 Virtual World (Second Life) Application

5.2.1 Interface

Next is the interface of the Virtual World Second Life composed by the objects that mimic the Real half of the project. In these images you can see the objects and dialog menus that are displayed when you interact with them.

5.2.2 The Second Life "House"

Figure 5.1: My Second Life "House" Overview

There following functions and state are used in all object scripts of the Virtual World part of the system so they'll be described first.

"llSay" is probably one of the most used functions and it enables "saying" something (the string) in the selected channel. This was taken from the sensor script in the Virtual Phidget so it is talking to the RFIDChannel, which is associated to him, telling to objects listening in that channel the tags it detected.

```
llSay(RFIDChannel,(string)tagsinList+detectedTags);
```

Listing 5.1: llsay method

And this is the function that starts the listening in the selected channel:

```
llListen(TagChannel,"","","");
```

Listing 5.2: lllisten method

In this case, it was also token from the sensor script in the Virtual Phidget, it listens on the "TagChannel".

And this is the state that is entered when "llListen" receives a message in the selected channel:

```
listen(integer channel, string name, key id, string msg)
```

Listing 5.3: listen state

The "listen" state receives the channel were the message was received, the name of the object/Avatar sending the message, his key and a string with the message.

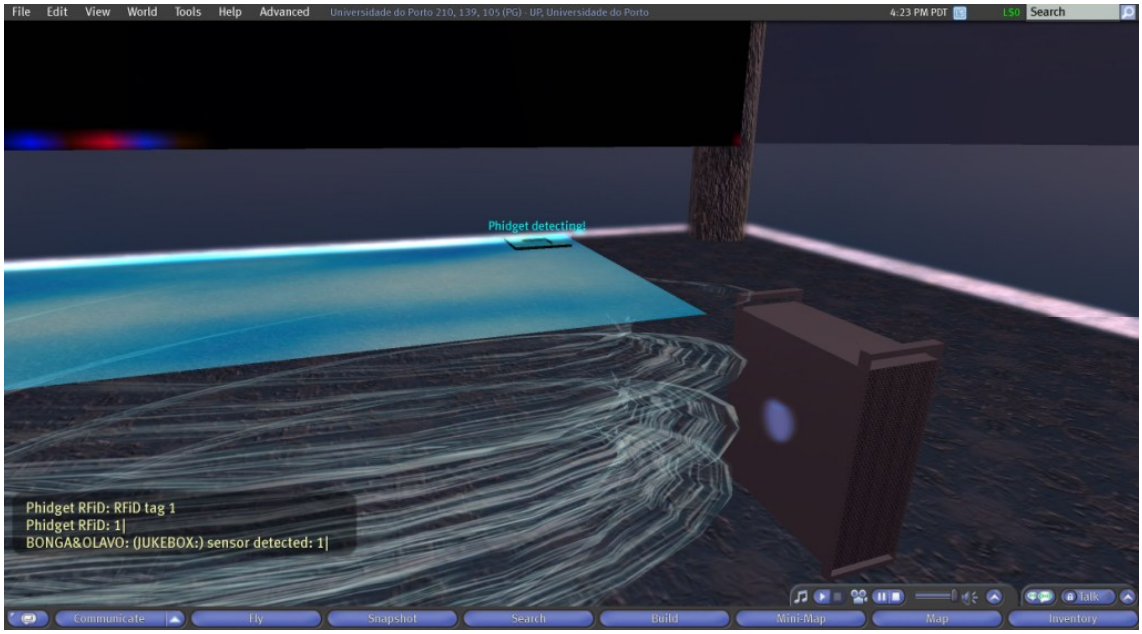

Figure 5.2: The Virtual Phidget RFID and the Virtual Server(G5)

5.2.3 The Virtual Phidget

The Virtual Phidget RFID contains the sensor script A.2 It's functioning is very simple, you touch it to connect it, and it starts searching for Avatars with:

```
llSensorRepeat("", "", AGENT, senseRange, senseArc, senseRate);
```

Listing 5.4: llsensorrepeat method

this can bring up two states: the "no_sensor" state, when nothing is detected, and the "sensor" state when it detects something (some AGENT in this case). When in "sensor" it starts sending information about Tags listened in the TagChannel. To stop the Virtual RFID touch it again.

5.2.4 Virtual Server

The Virtual Server object contains two scripts one is the HTTP Client A.4 the other a XML-RPC Server A.5.

The Client script sends whatever message(the string msg) it listens on the specified channel to the defined url:

```
http_request_id =llHTTPRequest(url, [HTTP_METHOD,"POST",HTTP_MIMETYPE,"text/plain;
    charset=utf-8"],msg);
```

Listing 5.5: llHTTPRequest method

and receives the answer:

```
http_response(key request_id, integer status, list metadata, string body)
```

Listing 5.6: http_response state

The XML-Server listens on the "remote_data" state:

```
remote_data(integer type, key channel, key message_id, string sender, integer ival,
    string sval)
```

Listing 5.7: remote_data state

And when receives something from outside Second Life processes the information and reply:

```
llRemoteDataReply(channel,NULL_KEY,"I got it",2008);
```

Listing 5.8: llRemoteDataReply method

5.2.5 The Video Player

Clicking the virtual display in the back will show this dialog menu. The script is a modified version of Freeview and allows Video-On-Demand.

Figure 5.3: Video Player

Only minor changes were made to the script so I can't take credit for it. See the full version here A.6.

5.2.6 The main MXR application: the Jukebox

Clicking the Jukebox object shows this dialog menu where you can see the playlist current entries, stop the currently playing music and choose another.

Figure 5.4: Jukebox: the central application in Second Life

The implemented Script in Jukebox A.1 object contains the most important part of the system the decider:

```
1  decider(integer channel,string msg)
2  {
3
4      if(channel == sensorChannel)
5      {
6          list parsed = llParseString2List(msg,["|","| ",","|"," | "],[]);
7          llOwnerSay("(JUKEBOX:) sensor detected: "+msg);
8          numberOfDetectedAvatars = llList2Integer(parsed,0);
9          if(numberOfDetectedAvatars==1)
10         {
11             DetectedAvatars+= [llList2String(parsed,2)];
12
13             if(llList2String(parsed,1)=="RFiD tag 1")
14                 presentVirtualState=1;
15             else presentVirtualState=2;
16         }
17         else if(numberOfDetectedAvatars>=2)
18             presentVirtualState=3;
19         else
20             presentVirtualState=0;
21     }
22     if(channel == serverChannel)
23     {
24         list parsed = llParseString2List(msg,["|","| ",","|"," | "],[]);
25         string type = llList2String(parsed,0);
```

```
26          llOwnerSay("Type: "+type);
27          if(type=="playURL")
28          {
29              tempurl  = llList2String(parsed,1);
30              llOwnerSay("url: "+tempurl);
31              musicName= llList2String(parsed,2);
32              changePlaylist(tempurl);//change the music url
33              return;
34          }
35          else if(type=="sensor")
36              presentRealState=llList2Integer(parsed,1);
37          tempurl  = llList2String(parsed,2);
38          musicName= llList2String(parsed,3);
39      }
40      if(presentVirtualState==0&&presentRealState==0)
41      {
42          tempurl="";
43          musicName="";
44          finalState=0;
45      }
46      else if(presentVirtualState==0)
47      {
48          //play the url sent from real world
49          finalState=presentRealState;
50      }
51      else if(presentRealState==0)
52      {
53          //play acording to virtual state
54          list parsedplaylist = llParseString2List(llList2String(playlist,(
                presentVirtualState-1)),["|","| ","| |"," | "],[]);
55          tempurl=llList2String(parsedplaylist,1);
56          musicName=llList2String(parsedplaylist,2);
57          finalState=presentVirtualState;
58      }
59      else
60      {
61          //the tricky part
62          if(presentVirtualState==1&&presentRealState==1||presentVirtualState==2&&
                presentRealState==2||presentVirtualState==3&&presentRealState==3)
63          {
64              list parsedplaylist = llParseString2List(llList2String(playlist,(
                    presentVirtualState-1)),["|","| ","| |"," | "],[]);
65              tempurl=llList2String(parsedplaylist,1);
66              musicName=llList2String(parsedplaylist,2);
67              finalState=presentVirtualState;
68          }
69          else
70          {
71              //for simplicity play the url 3 (index 2)
72              list parsedplaylist = llParseString2List(llList2String(playlist,2),["|"
                    ,"| ","| |"," | "],[]);
73              tempurl=llList2String(parsedplaylist,1);
74              musicName=llList2String(parsedplaylist,2);
75              finalState=3;
76
77          }
78      }
```

```
79
80      if(previousState!=finalState)
81          llSay(clientChannel,(string)finalState);
82      previousState=finalState;
83      changePlaylist(tempurl);//change the music url
84  }
```

Listing 5.9: Part of Jukebox LSL the decider() method

The outcome of this method is in the next table5.1. It shows the resulting URL depending on the users detected in both Worlds.

User Sensor (Phidget RFID) detection	Avatar Sensor(Virtual Phidget) detection	resulting URL
User 1	none	URL1
User 2	none	URL2
User 1 & 2	none	URL3
none	User1	URL1
none	User2	URL2
none	User1 & 2	URL3
User 1	User2	URL3
User 2	User1	URL3

Table 5.1: Decider method results

Clicking the next or previous button changes the music URL to play and you can either play it by pressing the “play” button or ignore the change pressing “ignore”:

Figure 5.5: Jukebox: Play

The script has more methods but for its relevance only “decider” was mentioned because most of the other LSL specific methods were already mentioned.

5.2.7 Virtual Tags

In this image you can see the two RFID Virtual Tags (the two rings over the table) and the Virtual Phidget RFID in the off State. The dialog menu appears when you touch one of the tags in this case Tag 1 and it asks your permission to attach the object.

Figure 5.6: The two RFID Virtual Tags

And the state that is triggered when you touch an object that has the touch start code in his script is this:

```
touch_start(integer num_detected) {
```

Listing 5.10: the touch start LSL state

Unlike other states the "touch start" does not need to be explicitly invoked it will run if you touch the object that contains it in its script.

Choosing "yes" will attach the Tag to your Avatar's left hand (if it was Tag 2 would attach to right hand).

This is the function that triggers the "run time permissions" state. It is inside the "touch start" so that it only happens when the object is touched.

```
llRequestPermissions(llDetectedKey(0), PERMISSION_ATTACH);
```

Listing 5.11: Run time permissions trigger LSL function

And this is the one that attach the object and it his located inside the "run time permissions" state:

```
llAttachToAvatar(ATTACH_LHAND);
```

Listing 5.12: Run time permissions trigger LSL function

Figure 5.7: Attaching the Tag to Avatar

The complete code is here A.3.

5.3 The iPlay Real World Support Application

5.3.1 iPlay interface

The main objectives to achieve with the interface were ease of use, self explaining buttons and control connectivity of some parts of the system. In the next image you can see the interface with the description of the buttons.

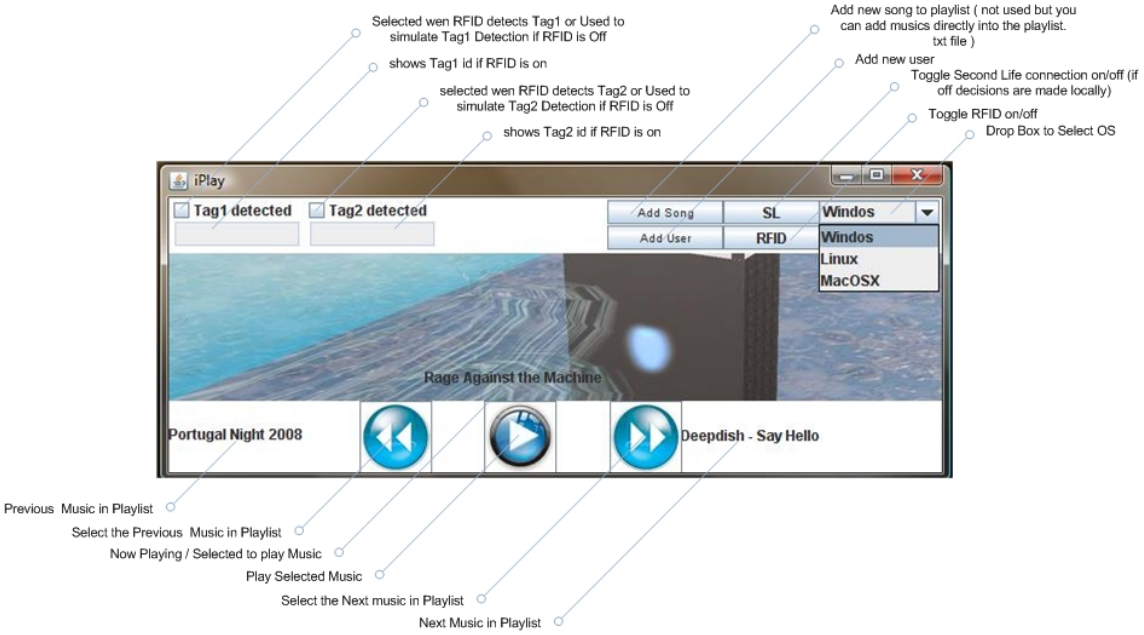

Figure 5.8: iPlay Interface (image captured in Windows Vista)

5.3.2 Design class diagrams

This is the implemented Classes diagram token from Eclipse using Omondo. As it shows the actually implemented classes don't necessarily match the conceptual ones 4.12.

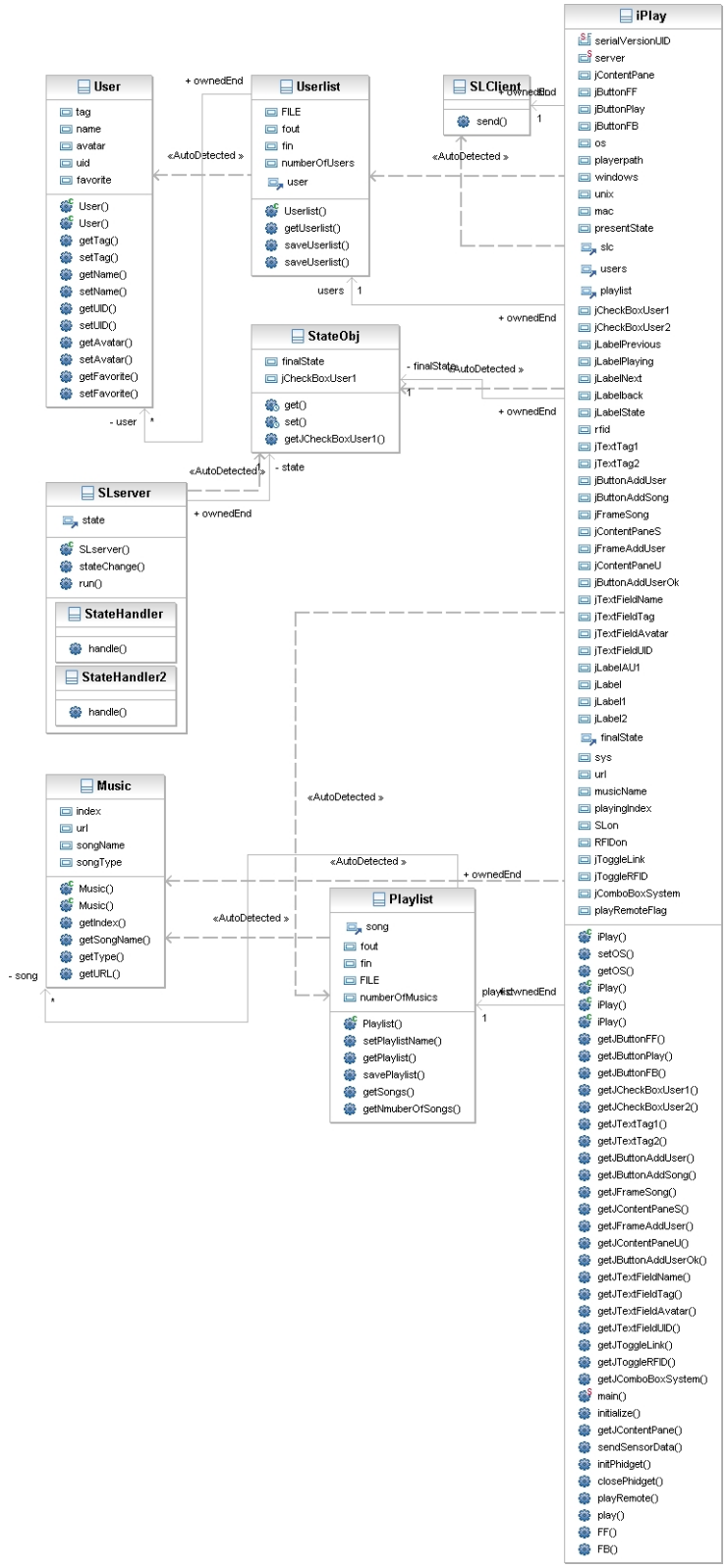

Figure 5.9: Design Class Diagram

5.3.3 Implementation Details

5.3.3.1 IPlay Implementation Details

The IPlay class B.1 is the central class of this implementation. It contains the main method as shown next and it implements the User Interface.

```
1   public static void main(String[] args){
2
3         final StateObj State=new StateObj();
4
5         (new Thread(new Runnable(){
6           public void run(){
7             new IPlay(State).setVisible(true);
8           }
9         }
10        )).start();
11
12
13        (server= new Thread(new SLserver(State))).start();
14
15  }
```

Listing 5.13: Threads launch in main( )

It launches two Threads the IPlay(interface) and the SLserver B.3(Second Life HTTP server) sharing the object StateObj B.2 between them so that information regarding Mixed-Reality State changes can be sent from the server to the interface. Initially the access to the shared object was made using a consumer/producer logic which sometimes froze the application (in the absence of response from Second Life) so to avoid deadlocks the application assumes the state in Real World as the Mixed-Reality (final) state and if/when response arrives from Second Life the state is updated.

It also starts the Phidget RFID using the method:

```
public void initPhidget()
```

Listing 5.14: initPhidget( )

using the methods from the Phidgets API:

```
1       rfid.addAttachListener(new AttachListener() {
2         public void attached(AttachEvent ae)
3         {
4           try
5           {
6             ((RFIDPhidget)ae.getSource()).setAntennaOn(true);
7             ((RFIDPhidget)ae.getSource()).setLEDOn(true);
8           }
9           catch (PhidgetException ex) { }
10          System.out.println("attachment of " + ae);
11        }
12      });
```

Listing 5.15: rfid.addAttachListener() tries to attach a Phidget RFID

These are only two of the details from the IPlay class but for their importance will be the only referred.

5.3.3.2 Playlist and Userlist Implementation Details

The Playlist B.8 and Userlist B.6 classes are in all aspects similar. When a new object from this classes is created the files Playlist.txt and Userlist.txt are read and an array of objects from the type Music B.7, for the Playlist, and from the type User B.5, for the Userlist, using retrieved information from the read files to fill up the objects field.

5.3.3.3 SLserver Implementation Details

The SLserver B.3 is a generic HTTP Server that receives incoming connections to the 80 pot and launches the designated Handler to the context if the context has one assigned. Then the handler by using the jCheckBox field from the StateObj signals the IPlay by setting the jCheckBox. The jCheckBox field was used because it already has an actionlistener associated to it and since the StateObj is shared between the two the listener will signal the IPlay that the StateObj has been changed. After that the IPlay gets the state field from the object.

5.3.3.4 SLClient Implementation Details

The SLClient B.4 implements a XML-RPC client used to send information to Second Life. In the line of creation of the object the Second Life XML-RPC server address must be given.

```
XmlRpcClient server = new XmlRpcClient("http://xmlrpc.secondlife.com/cgi-bin/
    xmlrpc.cgi");
```

Listing 5.16: XML-RPC detail

5.4 Conclusion

Second Life has no Database or any method of "physically" save the scripts states and variables. The most resembling usable in-world object is the notecard. Using notecards you can have in-world information but you can only read notecards you cannot write in them. So whenever a script is reset all variables state is lost.
Some positive features are the state-machine behavior of the LSL scripts which makes them very easy to understand and develop and the communications between in-world objects and/or avatars. Communicating is very easy just chose a channel and use the llSay function and objects or avatars listening in that channel will receive the sent message. A problem with this is that objects can not listen themselves so if you link some prims as the same object they won't be able to communicate with each other. Also if several scripts are running at the same time in the same parcel the Second Life Client will start running slower and slower. Another problem is the waiting time between uses some functions have. XML-RPC as 3sec waiting between consecutive use and HTTP requests 1sec.

Being made in Java iPlay runs in any OS that as installed the Java Virtual Machine. This aspect is also true for the Phidget RFID since drivers to all operating systems are available for download in the Phidgets page.

The Mixed-Reality system may not be using video as a response to the Mixed-Reality interactions but the support is there. The choice to use audio to test and validate the architecture was only because of the higher response times and the possibility of streaming over the internet without problems contrarily to video that requires much higher bandwidth and upstream(upload) speed.

Chapter 6

Use Case Realization

6.1 Introduction

In order to validate the architecture a model use case that involves all the components in the Mixed-Reality System in is realization will now be tested and described step by step.

6.2 The Mixed-Reality System at Work

Before starting the application make shore you open all the ports for both Darwin 3.10 and Second Life 3.7.1.

This is the Mixed-Reality system with all the components.

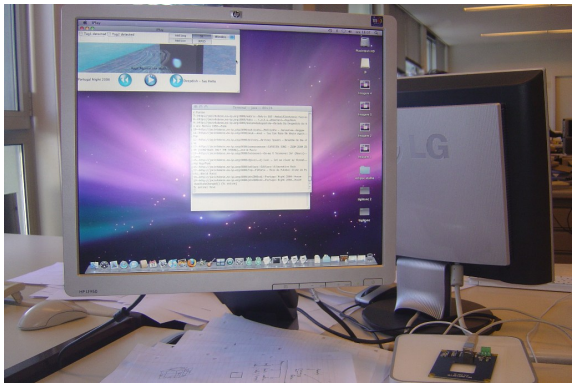

Figure 6.1: The Mixed-Reality System

The first step is to start the program using a terminal in super user mode (su). If you start the application by clicking the jar file to launch it, it will run but because of user privileges it won't be able to listen on the port 80.

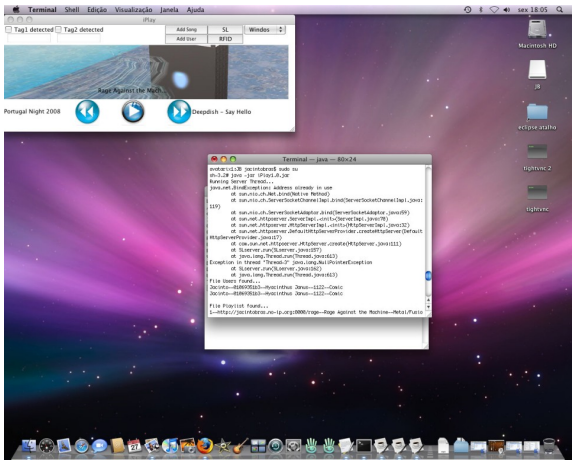

Figure 6.2: Starting the program

This is done with following command:

```
java -jar iPlay1.0.jar
```

Click the "SL" button to enable the program to send information to Second Life...

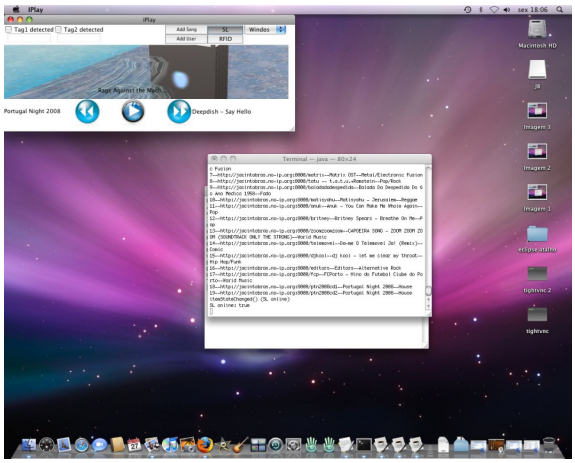

Figure 6.3: Enable information sending to Second Life

And the "RFID" button to "power on" the Phidget RFID. It was actually already connected but the antenna and the LED were off.

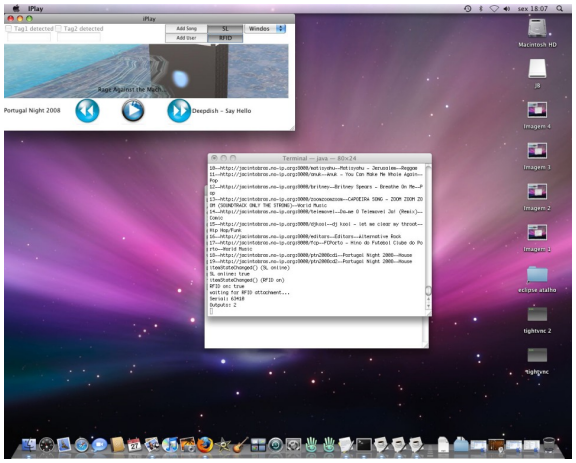

Figure 6.4: Connecting Phidget RFID

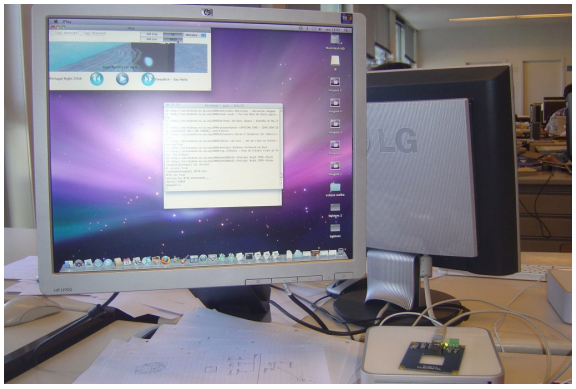

Figure 6.5: Phidget RFID connected

Now connect to Second Life.

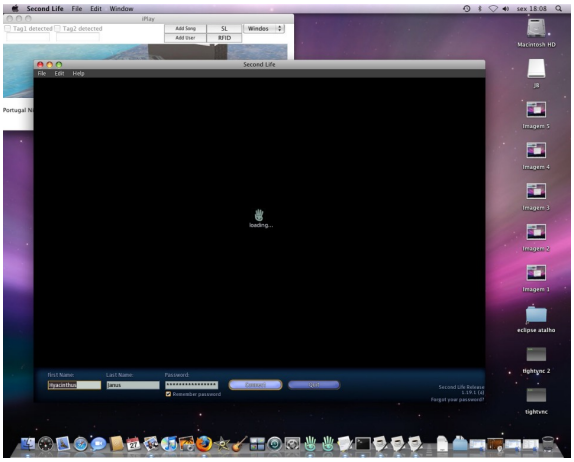

Figure 6.6: Starting the Second Life Client

Once in Second Life attach a tag to your avatar (Tag 1 in this case)...

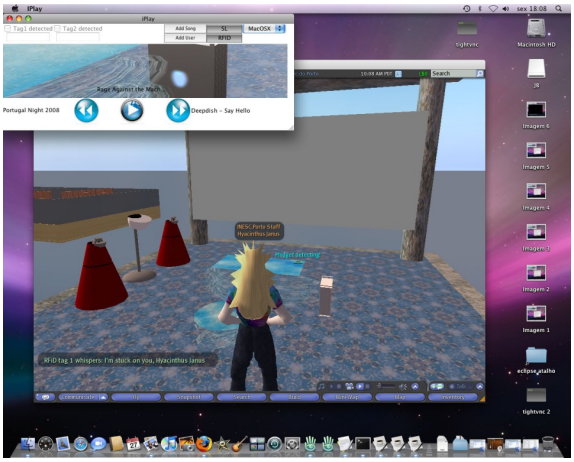

Figure 6.7: Tag attached

And connect the virtual RFID.

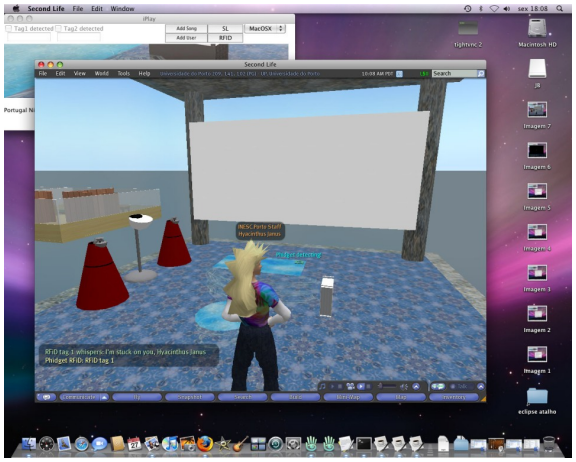

Figure 6.8: Virtual Phidget RFID connected

Now approach your avatar to the Virtual Phidget so it can be detected...

Figure 6.9: Avatar with tag 1 detected

And the Tag 2 in the Real World.

Figure 6.10: Tag 2 Detected and Video Playing

You can see the information that Tag 1 was detected in Second Life and Tag 2 was also detected in Real World.

Figure 6.11: Tag 1 and Tag 2 detected

Final result will be the song with URL3 being played in both Worlds.

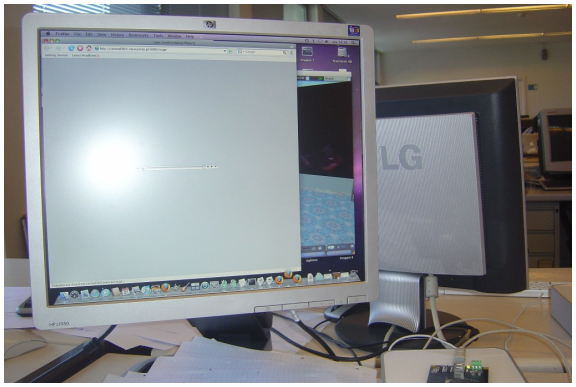

Figure 6.12: Music playing in Firefox

The resulting playing song is the one expect according to the table 5.1 showed in the previous chapter.

6.3 Conclusion

So by seeing the System response to the given stimulus and see it responds as was expected the architecture model was approved. Initially there were some problems in reaching the iPlay application with HTTP Requests from Second Life because of firewall blocking but the creation of a DMZ (DeMilitarized Zone) to the computer running it solved the problem.

Chapter 7

Conclusion

7.1 Summary

In the development of this thesys was tested and validated an architecture solution and a way to communicate between Real and Virtual World that makes possible Mixed-Reality interactions with content supported by the deployed system. These were the steps followed for the System's architecture deployment and its validation:

1. Streaming server deployment and testing;
2. sensing in Second Life deployment and testing;
3. Supporting Application in Second Life deployment and testing;
4. sensing in Real World integration and testing;
5. Supporting Application in Real World deployment and testing;
6. Mixed-Reality System testing.

Second Life client uses QuickTime plugins so the streaming must work in QuickTime Player or will not work on the Virtual world. Althoug support to video on demand was deployed the streaming of video works only correctly in LAN because of the bandwidth requirements wich made the choice of the interactive content to run the tests to be the music instead.

The Virtual World Second Life has no database so whenever a script is reset all variables state is lost making it impossible to maintain user data synchronized with real world data without having to use external tools to do so.

Also if several scripts are running at the same time in a parcel the Client will start running slower. Another problem is the waiting time between uses some functions induce to scripts. XML-RPC servers always delay received requests from outside for 3sec before sending the information to the waiting in-world object and also between consecutive HTTP

requests to the outside is a waiting period of 1sec. Because these methods are respectively the only possible ways in and out of the Virtual World there is no way to work around this problem and avoid the waiting times. Pay attention to the channel id the object waiting for outside information in Second Life is listening to because it must match the channel defined in the outside application. On other hand his very easy to communicate between in-world objects and/or avatars.

Firewall configuration is very important and if something is forgotten or badly configured you might not be capable of either reach inside Second Life or the support application outside or to have access to the audio and video feeds so make shore all the necessary ports are opened in the firewall and if necessary create a DMZ.

The support application iPlay and the Phidget RFID are OS independent. One problem with the Phidget RFID is that only one tag is detected at a time.

Overall the developed system architecture follows the initial Conceptual Class Model.

7.2 Results

To provide support for interactive content on a Mixed-Reality environment architecture and mechanism were developed for playing interactive multimedia content in Second Life and in the real world. This contribution is intended to make the experience of the Mixed-Reality User more fulfilling. Mixed-Reality context interactivity is supported, with video on demand support on both worlds. System responds to Mixed-Reality interactions bringing new functionalities in the Virtual and Real world. A larger object variety and higher level of interactivity, to Real Users and a more Real, immersive, response to Virtual Users interactions by manifesting the result in a real fashion with supporting hardware. So we take advantage of the best in the two worlds and combine it to create a Mixed-Reality world (environment) with all the advantages of both.

7.3 Future Work

Some of the more immediate improvements could be:

- Database access using HTTP requests to save users definitions and preferences;
- Install application in more computers with more RFID Readers;
- User context awareness – what media support the user disposes at a time;
- Play a specific content based on Avatar/user mood or preferences using the user profile and data crossing;
- Develop a Video-based application to detect user, user id and mood;
- Video Streaming of HD content over the internet;

- Create support for a more diverse multimedia content;
- Use Bluetooth, Wi-Fi or other technologies to interact with the system;
- User recognition trough video processing and face recognition.

To support improvements would be necessary the following add-ons:

- High Speed Uplink – with High Speed Internet access in both ways(uplink and downlink) like SDSL it would be possible to stream video in HD over the internet;
- Detect the media support devices available at a time;
- Adapt the media to the context of the user and to the available supporting devices;
- Add cameras and a fast computer for video processing to the System;
- Set up a DB such as MySQL or PostgreSQL to save system data.

The architecture, the provided interactive content and also the sensing solution might be used in future approaches to this same issue. If the Virtual World used is Second Life the communication must be done the same way at least in the near future because for now there is no other way. When mono[52] arrives to Second Life new possibilities might de opened and then the applications in both worlds could become very similar because mono is made in C#. This also gives new possibilities of interactions and perhaps adds new content support. If added to this the project of merging Second Life with Google Earth [45] ever became a reality we could even use GPS as a way to interact with the system using users and avatars positions to trigger the interactive content. In the future this could evolve to a highly interactive Mixed-Reality environment and with the advances in computer generated graphics achieve a state where users can't tell if they are in the virtual world or in the real world. Just like in the"Matrix"[53].

Appendix A

LSL

A.1 Jukebox Script

```
1  //JukeBox Script by Hyacinthus Janus
2
3  integer dialogChannel = 99;
4  integer serverChannel = 100;
5  integer sensorChannel = 103;
6  integer danceChannel = 102;
7  integer clientChannel= 110;
8  string nextCommand = "Next ";
9  string previousCommand = " Previous";
10 string playCommand = "Play >";
11 string stopCommand = "Stop []";
12 string playlistCommand = "Playlist [=]";
13 integer numberOfDetectedUsers=0;
14 list    DetectedUsers;
15 integer numberOfDetectedAvatars=0;
16 list    DetectedAvatars;
17 integer presentVirtualState=0;
18 integer presentRealState=0;
19 integer finalState=0;
20 integer previousState=0;
21 integer PRIM_GLOW = 25;
22 integer urlOffset=1;
23 integer nameOffset=2;
24 string NOTECARD = "urls";  //Used to host URL bookmarks for audio streams
25 key notecardkey = NULL_KEY;
26 key kQuery;
27 integer iLine = 0;
28 key tempuser;
29 string tempurl; //Temp string storge variable
30 string currenturl;
31 string musicName;
32 integer notecardcheck = 0;
33 integer numberofnotecardlines = 0;  //Stores the current number of detected
       notecard lines.
34 integer notecardline = 0;       //Current notecard line
35 integer isGroup = TRUE;
36 key groupcheck = NULL_KEY;
```

```
37
38 integer currentPlaylist = 0;
39
40 list playlist;//playlist readed from notecard
41
42 // Change the streaming URL
43 changePlaylist(string url)
44 {
45     llSetParcelMusicURL(url);//change the music url
46     llSetText(""+musicName+"\n",<1,.5,.5>,.3);
47 }
48 integer finditem(string name)   //Finds and returns an item s inventory number
49 {
50     integer i;
51     for(i=0;i<llGetInventoryNumber(INVENTORY_NOTECARD);i++)
52         if(llGetInventoryName(INVENTORY_NOTECARD,i) == NOTECARD)
53             return i;
54     return -1;
55 }
56 decider(integer channel,string msg)
57 {
58
59     if(channel == sensorChannel)
60     {
61         list parsed = llParseString2List(msg,["|","| "," |"," | "],[]);
62         llOwnerSay("(JUKEBOX:) sensor detected: "+msg);
63         numberOfDetectedAvatars = llList2Integer(parsed,0);
64         if(numberOfDetectedAvatars==1)
65         {
66             DetectedAvatars+= [llList2String(parsed,2)];
67
68             if(llList2String(parsed,1)=="RFiD tag 1")
69                 presentVirtualState=1;
70             else presentVirtualState=2;
71         }
72         else if(numberOfDetectedAvatars>=2)
73             presentVirtualState=3;
74         else
75             presentVirtualState=0;
76     }
77     if(channel == serverChannel)
78     {
79         list parsed = llParseString2List(msg,["|","| "," |"," | "],[]);
80         string type = llList2String(parsed,0);
81         llOwnerSay("Type: "+type);
82         if(type=="playURL")
83         {
84             tempurl  = llList2String(parsed,1);
85             llOwnerSay("url: "+tempurl);
86             musicName= llList2String(parsed,2);
87             changePlaylist(tempurl);//change the music url
88             return;
89         }
90         else if(type=="sensor")
91             presentRealState=llList2Integer(parsed,1);
92         tempurl  = llList2String(parsed,2);
93         musicName= llList2String(parsed,3);
```

```
94      }
95      if(presentVirtualState==0&&presentRealState==0)
96      {
97          tempurl="";
98          musicName="";
99          finalState=0;
100     }
101     else if(presentVirtualState==0)
102     {
103         //play the url sent from real world
104         finalState=presentRealState;
105     }
106     else if(presentRealState==0)
107     {
108         //play acording to virtual state
109         list parsedplaylist = llParseString2List(llList2String(playlist,(
                presentVirtualState-1)),["|","| ","  |"," | "],[]);
110         tempurl=llList2String(parsedplaylist,1);
111         musicName=llList2String(parsedplaylist,2);
112         finalState=presentVirtualState;
113     }
114     else
115     {
116         if(presentVirtualState==1&&presentRealState==1||presentVirtualState==2&&
                presentRealState==2||presentVirtualState==3&&presentRealState==3)
117         {
118             list parsedplaylist = llParseString2List(llList2String(playlist,(
                    presentVirtualState-1)),["|","| ","  |"," | "],[]);
119             tempurl=llList2String(parsedplaylist,1);
120             musicName=llList2String(parsedplaylist,2);
121             finalState=presentVirtualState;
122         }
123         else
124         {
125             //for simplicity play the url 3 (index 2)
126             list parsedplaylist = llParseString2List(llList2String(playlist,2),["|"
                    ,"| ","  |"," | "],[]);
127             tempurl=llList2String(parsedplaylist,1);
128             musicName=llList2String(parsedplaylist,2);
129             finalState=3;
130
131         }
132     }
133
134     if(previousState!=finalState)
135         llSay(clientChannel,(string)finalState);
136     previousState=finalState;
137     changePlaylist(tempurl);//change the music url
138 }
139 default
140 {
141     state_entry()
142     {
143         // listen for commands
144         llListen(dialogChannel, "", "", "");
145         llListen(serverChannel, "", "", "");
146         llListen(sensorChannel, "", "", "");
```

```
147         if(finditem(NOTECARD) != -1)
148             tempuser = llGetNumberOfNotecardLines(NOTECARD);
149         kQuery = llGetNotecardLine(NOTECARD, iLine);
150     }
151     on_rez(integer p)
152     {
153         llResetScript();
154     }
155     touch_start(integer num_detected)
156     {
157         llSetLinkPrimitiveParams(1,[ PRIM_GLOW, ALL_SIDES,  1.0 ]);
158         llDialog(llDetectedKey(0), "\n\nWould you like to change the playing music
                ?\n",
159             [previousCommand, stopCommand, nextCommand, playlistCommand],
                    dialogChannel);
160     }
161     listen(integer channel, string name, key id, string msg)
162     {
163         if(channel==dialogChannel)
164         {    //dialog box
165             if(msg == previousCommand)
166             {
167                 notecardline--;
168                 if(notecardline < 0)
169                     notecardline = numberofnotecardlines - 1;
170                 tempuser = id;
171                 llGetNotecardLine(NOTECARD,notecardline);
172             }
173             if(msg == nextCommand)
174             {
175                 notecardline++;
176                 if(notecardline >= numberofnotecardlines)
177                     notecardline = 0;
178                 tempuser = id;
179                 llGetNotecardLine(NOTECARD,notecardline);
180             }
181             if(msg == playCommand)
182             {
183                 changePlaylist(tempurl);
184             }
185             if(msg == stopCommand)
186             {
187                 changePlaylist("");
188             }
189             if(msg == playlistCommand)
190             {
191                 integer i=0;
192                 llOwnerSay("playlist:\n");
193                 while(llList2String(playlist,i)!="")
194                 {
195                     llOwnerSay("\t"+llList2String(playlist,i++));
196                 }
197             }
198         }
199         if(channel==serverChannel)
200         {   //server
201             decider(channel,msg);
```

```
202         }
203         if(channel==sensorChannel)
204         {   //sensor
205             decider(channel,msg);
206         }
207
208     }
209     changed(integer change)
210     {
211     if(change == CHANGED_INVENTORY) //If inventory change
212         if(finditem(NOTECARD) != -1)    // bookmarks notecard present
213             if(notecardkey != llGetInventoryKey(NOTECARD))
214                 tempuser = llGetNumberOfNotecardLines(NOTECARD); //Reload number of
                        lines
215     }
216     dataserver(key queryid, string data)
217     {
218         if(queryid == groupcheck)       //Test if object is deeded to group
219         {
220             groupcheck = NULL_KEY;
221             isGroup = FALSE;
222             return;
223         }
224         if (queryid == kQuery)
225         {
226             // this is a line of our notecard
227             if (data == EOF) {
228
229                 llSay(0, "No more lines in notecard, read " + (string)iLine + "
                        lines.");
230
231             } else {
232
233                 //request next line
234                 iLine++;
235                 kQuery = llGetNotecardLine(NOTECARD, iLine);   // read another line
                        when you can
236                 playlist = (playlist=[]) + playlist + [data];
237             }
238             return;
239         }
240         if(queryid == tempuser) //If just checking number of notecard lines
241         {
242             numberofnotecardlines = (integer)data;
243             notecardkey = llGetInventoryKey(NOTECARD);
244             notecardcheck = 0;
245             llGetNotecardLine(NOTECARD, notecardcheck);
246             return;
247         }
248         if(notecardcheck != -1 )
249         {
250             if(data != EOF)
251             {
252                 if(data == "")
253                 {
254                     notecardcheck++;
255                     llGetNotecardLine(NOTECARD, notecardcheck);
```

```
256                 }
257                 else
258                 {
259                     notecardcheck = -1;
260                     return;
261                 }
262             }
263             else
264                 return;
265         }
266         if(data == "" && notecardline < numberofnotecardlines )     //If user just
              pressed "enter" in bookmarks, skip
267         {
268             notecardline++;
269             llGetNotecardLine(NOTECARD, notecardline);
270             return;
271         }
272         if(data == EOF)
273         {
274             notecardline = 0;
275             llGetNotecardLine(NOTECARD, notecardline);
276             return;
277         }
278         list parsed = llParseString2List(data,["|","| "," |"," | "],[]);     //
              Ensure no blank spaces before "http://".
279         string number = llList2String(parsed,0);
280         tempurl = llList2String(parsed,1);
281         musicName = llList2String(parsed,2);
282         if(tempurl == "")
283             tempurl = "** No URL specified! **";
284         //llOwnerSay(tempurl);
285         llDialog(tempuser,"Jukebox playlists ("+(string)(notecardline+1)+"/"+(
              string)numberofnotecardlines+")\n"+number+" - ("+musicName+")\n"+
              tempurl,[previousCommand, playCommand, nextCommand],dialogChannel);
286
287 }
288     }
```

Listing A.1: Jukebox LSL

A.2 Sensor Script

```
1  //Sensor Script by Hyacinthus Janus
2
3  float senseRange = 3.0; // search within 3m
4  float senseArc = PI;      // search around the object
5  float senseRate = 10.0;    // search every 10 seconds
6  string AllAgents;
7
8  string detectedTags;
9  string TAGDETECTED;
10 integer previousnumberDetected=0;
11 //integer sensorChannel = 101;
12 integer RFIDChannel = 103;
13 integer TagChannel = 105;
14 list TagsDetectedIDs;
15 integer tagsinList=0;
16
17
18 default
19 {
20     state_entry()
21     {
22         // det the sensor to search for Tags
23
24         llSetText("Touch me! I'm a Phidget!",<0,1,1>,5);
25
26     }
27     on_rez(integer p)
28     {
29         llResetScript();
30     }
31     touch_start(integer total_number)
32     {
33         llSetText("Phidget is going to detect...",<0,1,1>,5);
34         state detect_on;
35     }
36     state_exit()
37     {
38
39     }
40
41 }
42
43 state detect_on
44 {
45     state_entry()
46     {
47         // det the sensor to search for Tags
48         llSensorRepeat("", "", AGENT, senseRange, senseArc, senseRate);
49         llListen(TagChannel,"","","");
50         llSetText("Phidget detecting!",<0,1,1>,5);
51         previousnumberDetected=0;
52     }
53     no_sensor()
54     {
55         if(previousnumberDetected>0)
```

```
56          {
57              llOwnerSay("Nothing Sensed");
58          }
59          previousnumberDetected=0;
60
61
62      }
63      sensor (integer numberDetected)
64      {
65          //string thisTag = "";
66          integer thisNum;
67          integer agentsDetected=0;
68          integer tagsDetected=0;
69           detectedTags="";
70           list detected;
71          //state change
72          for (thisNum=0; thisNum<numberDetected; thisNum++)
73          {
74              key thisKey = llDetectedKey(thisNum);
75              string thisAgent = llDetectedName(thisNum);
76
77              if(previousnumberDetected!=numberDetected)
78              {
79                  llOwnerSay("Sensed "+llDetectedName(thisNum)+" at "+(string)
                        llDetectedPos(thisNum));
80                  detected = (detected=[]) + detected + [(string)thisKey+"|"+
                        thisAgent];
81                  agentsDetected++;
82              }
83
84
85             // llOwnerSay("Sensed "+llDetectedName(TagNum)+" at "+(string)
                   llDetectedPos(TagNum));
86          }
87
88
89          integer i=(tagsinList-1);
90          while(i>-1)
91              detectedTags+= "|"+llList2String(TagsDetectedIDs,i--);
92          //tagsDetected
93          llOwnerSay  ((string)tagsinList+detectedTags);
94          llSay(RFIDChannel,(string)tagsinList+detectedTags);
95          //llOwnerSay((string)tagsDetected+"|"+detectedTags);
96          previousnumberDetected=numberDetected;
97          TagsDetectedIDs = llListReplaceList(TagsDetectedIDs, [""], 0,
                llGetListLength(TagsDetectedIDs));
98          tagsinList=0;
99      }
100     listen(integer channel, string name, key id, string msg)
101     {
102         if(channel==TagChannel)
103         {
104             //llOwnerSay(msg);
105             if(llGetSubString(msg,0,3) == "RFiD")
106             {
107
108                 list parsed = llParseString2List(msg,["|","| "," |"," | "],[]);
```

```
109                 //llOwnerSay(llList2String(parsed,1));
110         llSetTimerEvent(10);//10s
111                 //if(previousnumberDetected==0)
112                 TAGDETECTED = llList2String(parsed,1);
113                 integer index = llListFindList( TagsDetectedIDs, [llList2String(
                        parsed,1)]);
114                 if(index==-1)//add new tag id to the list
115                 {
116                     llOwnerSay(llList2String(parsed,0));
117                     TagsDetectedIDs = (TagsDetectedIDs=[]) + TagsDetectedIDs + [
                            llList2String(parsed,1)];
118                     TagsDetectedIDs = (TagsDetectedIDs=[]) + TagsDetectedIDs + [
                            llList2String(parsed,0)];
119                     tagsinList++;
120                 }
121                 //llSay(RFIDChannel,(string)tagsDetected+"|"+detectedTags);
122                 //lSay(RFIDChannel,);
123
124
125             }
126         }
127
128     }
129     touch_start(integer total_number)
130     {
131         llSetText("Phidget is going off...",<0,1,1>,5);
132         state default;
133     }
134 }
```

Listing A.2: Sensor LSL

A.3 Tag Script

```
1  //Tag1 Script by Hyacinthus Janus
2
3  integer TagChannel = 104;
4
5  default {
6
7      state_entry()
8      {
9          llSetText("Tag 1!\n Touch me to attach...",<.8,.8,0>,0.8);
10     }
11     touch_start(integer num_detected) {
12
13         llRequestPermissions(llDetectedKey(0), PERMISSION_ATTACH);
14         if(llGetAttached())
15             llSetPos(<0, 0, 0>);
16     }
17
18     run_time_permissions(integer perm) {
19         if (perm & PERMISSION_ATTACH)  {
20             llAttachToAvatar(ATTACH_LHAND);
21         }
22     }
23     attach(key attached)
24     {
25         if (attached == NULL_KEY)  // object has been detached
26         {
27             llWhisper( 0, "Why hast thou forsaken me?" );
28             llSetPos(<209.933, 138.733, 106.573>);
29             llSetText("",<.5,.5,0>,0.5);
30       llSetTimerEvent(0);//stop timer
31         }
32         else   // object has been attached//
33         {
34             llWhisper( 0, "I'm stuck on you, " + llKey2Name(attached) );
35             llSetTimerEvent(10);//10s
36             llSetText("",<.5,.5,0>,0.5);
37         }
38     }
39     timer()
40     {
41         llSay(TagChannel,"RFiD tag 1"+"|"+(string)llGetKey());
42     }
43
44 }
```

Listing A.3: Tag1 LSL

A.4 Client Script

```
1  //Client Script by Hyacinthus Janus
2
3  key http_request_id;
4  string url="http://secondlife5.inescporto.pt/state";
5  integer clientChannel=110;
6
7  default
8  {
9      state_entry()
10     {
11         llListen(clientChannel,"","","");
12         llSetText("", <0,0,1>, 1);
13     }
14     listen(integer channel, string name, key id, string msg)
15     {
16         if(channel==clientChannel)
17         {
18
19             llOwnerSay("Message to send to http server  "+msg);
20             http_request_id =llHTTPRequest(url, [HTTP_METHOD,"POST",HTTP_MIMETYPE,
                   "text/plain;charset=utf-8"],msg);
21         }
22     }
23     http_response(key request_id, integer status, list metadata, string body)
24     {
25         if (request_id == http_request_id)
26         {
27             llOwnerSay("Status:"+(string)status+" Meta: "+(string)metadata);
28             llSetText(body, <0,0,1>, 1);
29         }
30     }
31 }
```

Listing A.4: Client LSL

A.5 Server Script

```
1  //Server Script by Hyacinthus Janus
2
3  integer serverChannel=100;
4
5  key remoteChannel;
6
7  init() {
8      llOpenRemoteDataChannel(); // create an XML-RPC channel
9      llOwnerSay("My key is " + (string)llGetKey());
10 }
11
12 default {
13     state_entry() {
14         init();
15     }
16
17     state_exit() {
18         return;
19     }
20
21     on_rez(integer param) {
22         llResetScript();
23     }
24
25     remote_data(integer type, key channel, key message_id, string sender, integer
           ival, string sval) {
26          if (type == REMOTE_DATA_CHANNEL) { // channel created
27              llSay(DEBUG_CHANNEL,"Channel opened for REMOTE_DATA_CHANNEL" +
28                 (string)channel + " " + (string)message_id + " " + (string)sender +
                       " " +
29                 (string)ival + " " + (string)sval);
30              remoteChannel = channel;
31              llOwnerSay("Ready to receive requests on channel \"" + (string)channel
                     + "\"");
32              state receiving; // start handling requests
33          } else {
34              llSay(DEBUG_CHANNEL,"Unexpected event type");
35          }
36      }
37 }
38
39
40 state receiving {
41
42     state_entry() {
43         llOwnerSay("Ready to receive information from outside SL");
44     }
45
46     state_exit() {
47         llOwnerSay("No longer receiving information from outside SL.");
48         llCloseRemoteDataChannel(remoteChannel);
49     }
50
51     on_rez(integer param) {
52         llResetScript();
```

```
53      }
54
55      remote_data(integer type, key channel, key message_id, string sender, integer
            ival, string sval) {
56          if (type == REMOTE_DATA_REQUEST) { // handle requests sent to us
57              llSay(DEBUG_CHANNEL,"Request received for REMOTE_DATA_REQUEST " + (
                    string)channel + " " +
58                  (string)message_id + " " + (string)sender + " " + (string)ival + "
                        " + (string)sval);
59              llRemoteDataReply(channel,NULL_KEY,"I got it",2008);
60              llOwnerSay("I just received data in "+ llGetRegionName() +
61                          " at position " + (string)llGetPos() + "\n" +
62                         "The string was " +  sval + "\nThe number was " + (string)
                            ival + ".");
63          }
64          llSay(serverChannel,sval);
65      }
66
67  }
```

Listing A.5: Server LSL

A.6 FreeView 1.2 Script

[54]

```
1 //FreeView 1.2 WebGuide - By CrystalShard Foo
2 //Multifunctional Picture viewer and Video control script with webguide support
3 //This script is distributed for free and must stay that way. DO NOT SELL THIS
      SCRIPT UNDER ANY CIRCUMSTANCE.
4
5 //Help for using this script can be obtained at: http://www.slguide.com/help
6
7 //Feel free to modify this script and post your improvement. Leave the credits
      intact but feel free to add your name at its bottom.
8
9
10 //Constants
11 integer PICTURE_ROTATION_TIMER = 30;    //In whole seconds
12
13 integer DISPLAY_ON_SIDE = ALL_SIDES; //Change this to change where the image will
      be displayed
14
15 key VIDEO_DEFAULT = "6e0f05ad-1809-4edc-df29-fae3d2a6c9b8";  //Test pattern - Used
      as default video texture when one is missing in parcel media
16 //key BLANK = "5748decc-f629-461c-9a36-a35a221fe21f"; //Blank texture - Used when
      there are no textures to display in Picture mode
17 key BLANK = "8b5fec65-8d8d-9dc5-cda8-8fdf2716e361";
18
19 string NOTECARD = "bookmarks";  //Used to host URL bookmarks for video streams
20
21 integer VIDEO_MATERIAL = PRIM_MATERIAL_LIGHT;     //Note - Side 0 automaticly gets
      colored <0,0,0> to prevent light lag!
22 integer PICTURE_MATERIAL = PRIM_MATERIAL_LIGHT;  //All this does is make the prim
      bright. No worries about light related lag.
23
24 integer REMOTE_CHANNEL = 9238742;
25
26 integer mode = 0;            //Freeview mode.
27                               //Mode 0 - Power off
28                               //Mode 1 - Picture viewer
29                               //Mode 2 - Video
30
31 integer listenHandle = -1;       //Dialog menu listen handler
32 integer listenUrl = -1;          //listen handler for channel 1 for when a URL is
      being added
33 integer listenTimer = -1;        //Timer variable for removing all listeners after 2
       minutes of listener inactivity
34 integer listenRemote = -1;       //listen handler for the remote during initial
      setup
35 integer encryption = 0;
36 integer numberofnotecardlines = 0;  //Stores the current number of detected
      notecard lines.
37 integer notecardline = 0;        //Current notecard line
38
39 integer loop_image = FALSE;      //Are we looping pictures with a timer? (picture
      mode)
40 integer current_texture = 0;     //Current texture number in inventory being
      displayed (picture mode)
```

```
41 integer chan;                    //llDialog listen channel
42 integer notecardcheck = 0;
43 key video_texture;               //Currently used video display texture for parcel
       media stream
44
45 string moviename;
46 string tempmoviename;
47 key notecardkey = NULL_KEY;
48 key tempuser;                    //Temp key storge variable
49 string tempurl;                  //Temp string storge variable
50
51 integer isGroup = TRUE;
52 key groupcheck = NULL_KEY;
53 key last_owner;
54 key XML_channel;
55
56 pictures()       //Change mode to Picture Viewer
57 {
58     //Initilize variables
59
60     //Change prim to Light material while coloring face 0 black to prevent light-
           lag generation.
61     llSetPrimitiveParams([PRIM_BUMP_SHINY,DISPLAY_ON_SIDE,PRIM_SHINY_NONE,
           PRIM_BUMP_NONE,PRIM_COLOR,DISPLAY_ON_SIDE,<1,1,1>,1.0,PRIM_MATERIAL,
           PICTURE_MATERIAL]);
62     llSetColor(<0,0,0>,0);
63
64     integer check = llGetInventoryNumber(INVENTORY_TEXTURE);
65
66     if(check == 0)
67     {
68         report("No pictures found.");
69         llSetTexture(BLANK,DISPLAY_ON_SIDE);
70         return;
71     }
72     else
73         if(current_texture > check)
74             //Set to first texture if available
75             current_texture = 0;
76
77     display_texture(current_texture);
78 }
79
80 video()          //Change mode to Video
81 {
82     //Change prim to Light material while coloring face 0 black to prevent light-
           lag generation.
83     llSetPrimitiveParams([PRIM_BUMP_SHINY,DISPLAY_ON_SIDE,PRIM_SHINY_NONE,
           PRIM_BUMP_NONE,PRIM_COLOR,DISPLAY_ON_SIDE,<1,1,1>,1.0,PRIM_MATERIAL,
           VIDEO_MATERIAL,PRIM_TEXTURE,DISPLAY_ON_SIDE,"62dc73ca-265f-7ca0-0453-
           e2a6aa60bb6f",llGetTextureScale(DISPLAY_ON_SIDE),llGetTextureOffset(
           DISPLAY_ON_SIDE),llGetTextureRot(DISPLAY_ON_SIDE)]);
84     llSetColor(<0,0,0>,0);
85
86     report("Video mode"+moviename+": Stopped");
87     if(finditem(NOTECARD) != -1)
88         tempuser = llGetNumberOfNotecardLines(NOTECARD);
```

```
89      video_texture = llList2Key(llParcelMediaQuery([PARCEL_MEDIA_COMMAND_TEXTURE])
            ,0);
90      if(video_texture == NULL_KEY)
91      {
92          video_texture = VIDEO_DEFAULT;
93          llParcelMediaCommandList([PARCEL_MEDIA_COMMAND_TEXTURE, VIDEO_DEFAULT]);
94          llSay(0,"No parcel media texture found. Setting texture to default: "+(
                string)VIDEO_DEFAULT);
95          if(llGetLandOwnerAt(llGetPos()) != llGetOwner())
96              llSay(0,"Error: Cannot modify parcel media settings. "+llGetObjectName
                    ()+" is not owned by parcel owner.");
97      }
98
99      llSetTexture(video_texture,DISPLAY_ON_SIDE);
100 }
101
102 off()
103 {
104     report("Click to power on.");
105     llSetPrimitiveParams([PRIM_BUMP_SHINY,DISPLAY_ON_SIDE,PRIM_SHINY_LOW,
            PRIM_BUMP_NONE,PRIM_COLOR,DISPLAY_ON_SIDE,<0.1,0.1,0.1>,1.0,PRIM_MATERIAL,
            PRIM_MATERIAL_PLASTIC,PRIM_TEXTURE,DISPLAY_ON_SIDE,BLANK,llGetTextureScale(
            DISPLAY_ON_SIDE),llGetTextureOffset(DISPLAY_ON_SIDE),llGetTextureRot(
            DISPLAY_ON_SIDE)]);
106 }
107
108 integer finditem(string name)    //Finds and returns an item's inventory number
109 {
110     integer i;
111     for(i=0;i<llGetInventoryNumber(INVENTORY_NOTECARD);i++)
112         if(llGetInventoryName(INVENTORY_NOTECARD,i) == NOTECARD)
113             return i;
114     return -1;
115 }
116
117 seturl(string url, key id)  //Set parcel media URL
118 {
119     if(mode != 2)
120     {
121         video();
122         mode = 2;
123     }
124     moviename = tempmoviename;
125     if(moviename)
126         moviename = " ["+moviename+"]";
127     tempmoviename = "";
128     string oldurl = llList2String(llParcelMediaQuery([PARCEL_MEDIA_COMMAND_URL]),0)
            ;
129     if(oldurl != "")
130         llOwnerSay("Setting new media URL. The old URL was: "+oldurl);
131
132     llParcelMediaCommandList([PARCEL_MEDIA_COMMAND_URL, url]);
133     if(id!=NULL_KEY)
134         menu(id);
135     else
136     {
137         report("Video mode"+moviename+": Playing");
```

```
138         llParcelMediaCommandList([PARCEL_MEDIA_COMMAND_PLAY]);
139     }
140
141     if(isGroup)
142         llSay(0,"New media URL set.");
143     else
144         llOwnerSay("New media URL set: "+url);
145 }
146
147 string mediatype(string ext)    //Returns a string stating the filetype of a file
        based on file extension
148 {
149     ext = llToLower(ext);
150     if(ext == "swf")
151         return "Flash";
152     if(ext == "mov" || ext == "mp4" || ext == "avi" || ext == "mpg" || ext == "mpeg
            " || ext == "smil")
153         return "Video";
154     if(ext == "jpg" || ext == "mpeg" || ext == "gif" || ext == "png" || ext == "
            pict" || ext == "tga" || ext == "tiff" || ext == "sgi" || ext == "bmp")
155         return "Image";
156     if(ext == "txt")
157         return "Text";
158     return "Unknown";
159 }
160
161 browse(key id)      //Image browser function for picture viewer mode
162 {
163     integer check = llGetInventoryNumber(INVENTORY_TEXTURE);
164     string header;
165     if(check > 0)
166         header = "("+(string)(current_texture+1)+"/"+(string)check+") "+
                llGetInventoryName(INVENTORY_TEXTURE,current_texture);
167     else
168         header = "No pictures found.";
169     llDialog(id,"** Monitor Control **\n Picture Viewer mode\n- Image browser\n- "+
            header,["Back","Next","Menu"],chan);
170     extendtimer();
171 }
172
173 report(string str)
174 {
175     llSetObjectDesc(str);
176 }
177
178 extendtimer()       //Add another 2 minute to the Listen Removal timer (use when a
        Listen event is triggered)
179 {
180     if(listenHandle == -1)
181         listenHandle = llListen(chan,"","","");
182     listenTimer = (integer)llGetTime() + 120;
183     if(loop_image == FALSE)
184         llSetTimerEvent(45);
185 }
186
187 config(key id)      //Configuration menu
188 {
```

```
189     extendtimer();
190     llDialog(id,"Current media URL:\n"+llList2String(llParcelMediaQuery([
            PARCEL_MEDIA_COMMAND_URL]),0)+"\nTip: If the picture is abit off, try '
            Align ON'",["Set URL","Align ON","Align OFF","Menu","Set Remote"],chan);
191 }
192
193 tell_remote(string str)
194 {
195     llShout(REMOTE_CHANNEL, llXorBase64Strings(llStringToBase64((string)encryption +
            str), llStringToBase64((string)encryption)));
196 }
197
198 menu(key id)          //Dialog menus for all 3 modes
199 {
200     list buttons = [];
201     string title = "** Monitor control **";
202
203     extendtimer();
204
205     if(mode != 0)
206     {
207         if(mode == 1)        //Pictures menu
208         {
209             title+="\n  Picture Viewer mode";
210             buttons+=["Browse"];
211             if(loop_image == FALSE)
212                 buttons+=["Loop"];
213             else
214                 buttons+=["Unloop"];
215             buttons+=["Video","Power off","Help"];
216         }
217         else                 //Video menu
218         {
219             title+="\n Video display mode\n"+moviename+"\nTip:\nClick 'Web Guide'
                    to view the Online bookmarks.";
220             buttons+=["Pictures","Configure","Power off","Loop","Unload","Help","
                    Play","Stop","Pause","Web Guide","Bookmarks","Set URL"];
221         }
222     }
223     else
224         buttons += ["Pictures","Video","Help"];
225
226     llDialog(id,title,buttons,chan);
227 }
228
229 display_texture(integer check)  //Display texture and set name in description (
        picture mode)
230 {                               //"Check" holds the number of textures in contents.
        The function uses "current_texture" to display.
231     string name = llGetInventoryName(INVENTORY_TEXTURE, current_texture);
232     llSetTexture(name,DISPLAY_ON_SIDE);
233     report("Showing picture: "+name+" ("+(string)(current_texture+1)+"/"+(string)
            check+")");
234 }
235
236
237 next()  //Change to next texture (picture mode)
```

```
238 {       //This function is used twice - by the menu and timer. Therefor, it is a
        dedicated function.
239     current_texture++;
240     integer check = llGetInventoryNumber(INVENTORY_TEXTURE);
241     if(check == 0)
242     {
243         llSetTexture(BLANK,DISPLAY_ON_SIDE);
244         current_texture = 0;
245         report("No pictures found.");
246         return;
247     }
248     if(check == current_texture)
249         current_texture = 0;
250
251     display_texture(check);
252     return;
253 }
254
255 default
256 {
257     state_entry()
258     {
259         chan = (integer)llFrand(1000) + 1000;    //Pick a random listen channel for
                the listener
260         if(PICTURE_ROTATION_TIMER <= 0)          //Ensure the value is no less or
                equal 0
261             PICTURE_ROTATION_TIMER = 1;
262         llListenRemove(listenHandle);
263         listenHandle = -1;
264         last_owner = llGetOwner();
265         groupcheck = llRequestAgentData(llGetOwner(),DATA_NAME);
266         off();
267         llOpenRemoteDataChannel();
268     }
269
270     on_rez(integer i)
271     {
272         llResetScript();
273     }
274
275     touch_start(integer total_number)
276     {
277         //
                ------------------------------------------------------------------------

278         //Listen only to owner or group member. Edit this code to change access
                controls.
279         if(llDetectedKey(0) != llGetOwner() && llDetectedGroup(0) == FALSE)
280             return;
281         //
                ------------------------------------------------------------------------

282
283         if(llGetOwnerKey(llGetKey()) != last_owner)  //Sense if object has been
                deeded to group for Web Guide function
284         {
285             isGroup = TRUE;
```

```
286             last_owner = llGetOwner();
287             groupcheck = llRequestAgentData(llGetOwner(),DATA_NAME);
288         }
289 
290         menu(llDetectedKey(0));
291     }
292 
293     changed(integer change)
294     {
295         if(change == CHANGED_INVENTORY) //If inventory change
296             if(mode == 1)    //If picture mode
297             {
298                 integer check = llGetInventoryNumber(INVENTORY_TEXTURE);
299                 if(check != 0)
300                 {
301                     current_texture = 0;
302                     display_texture(check);
303                 }
304                 else
305                 {
306                     llSetTexture(BLANK,DISPLAY_ON_SIDE);
307                     report("No pictures found.");
308                 }
309             }
310             else
311                 if(mode == 2)    //If video mode
312                     if(finditem(NOTECARD) != -1)     //And bookmarks notecard
                            present
313                         if(notecardkey != llGetInventoryKey(NOTECARD))
314                             tempuser = llGetNumberOfNotecardLines(NOTECARD);     //
                                    Reload number of lines
315     }
316 
317     listen(integer channel, string name, key id, string message)
318     {
319         if(message == "Pictures")
320         {
321             if(mode == 2)
322                 llParcelMediaCommandList([PARCEL_MEDIA_COMMAND_STOP]);
323             pictures();
324             mode = 1;
325             menu(id);
326             return;
327         }
328         if(message == "Video")
329         {
330             video();
331             mode = 2;
332             menu(id);
333             return;
334         }
335         if(message == "Power off")
336         {
337             if(mode == 2)
338                 llParcelMediaCommandList([PARCEL_MEDIA_COMMAND_UNLOAD]);
339             off();
340             mode = 0;
```

```
341             return;
342         }
343         if(message == "Help")
344         {
345             llSay(0,"Help documentation is available at: http://www.slguide.com/
                    help");
346             if(isGroup)
347                 tell_remote("HELP"+(string)id+(string)XML_channel);
348             else
349                 llLoadURL(id,"Help pages for FreeView","http://www.slguide.com?c="
                        +(string)XML_channel+"&help=1");
350         }
351         if(mode == 1)
352         {
353             if(message == "Browse")
354             {
355                 loop_image = FALSE;
356                 browse(id);
357                 return;
358             }
359             if(message == "Next")
360             {
361                 extendtimer();
362                 next();
363                 browse(id);
364             }
365             if(message == "Back")
366             {
367                 extendtimer();
368                 current_texture--;
369                 integer check = llGetInventoryNumber(INVENTORY_TEXTURE);
370                 if(check == 0)
371                 {
372                     llSetTexture(BLANK,DISPLAY_ON_SIDE);
373                     current_texture = 0;
374                     report("No pictures found.");
375                     return;
376                 }
377                 if(current_texture < 0)
378                     current_texture = check - 1;
379
380                 display_texture(check);
381
382                 browse(id);
383                 return;
384             }
385             if(message == "Menu")
386             {
387                 menu(id);
388                 return;
389             }
390             if(message == "Loop")
391             {
392                 llSetTimerEvent(PICTURE_ROTATION_TIMER);
393                 loop_image = TRUE;
394                 llOwnerSay("Picture will change every "+(string)
                        PICTURE_ROTATION_TIMER+" seconds.");
```

```
395                 return;
396             }
397             if(message == "Unloop")
398             {
399                 loop_image = FALSE;
400                 llOwnerSay("Picture loop disabled.");
401                 return;
402             }
403         }
404         if(mode == 2)
405         {
406             if(channel == REMOTE_CHANNEL)
407             {
408                 if(encryption == 0)
409                     encryption = (integer)message;
410                 llListenRemove(listenRemote);
411                 listenRemote = -1;
412                 llSay(0,"Remote configured ("+(string)id+")");
413             }
414 
415             if(message == "Web Guide")
416             {
417                 if(isGroup)
418                 {
419                     if(!encryption)
420                     {
421                         llSay(0,"** Error - This FreeView object has been deeded to
                                group. You must use a Remote control to open the Web
                                Guide.");
422                         llSay(0,"You can set up the remote control from the Video
                                -> Configuration menu. Please refer to the notecard for
                                further assistance.");
423                         return;
424                     }
425                     tell_remote((string)id+(string)XML_channel+(string  )llGetOwner
                            ());
426                 }
427                 else
428                     llLoadURL(id, "Come to the Guide to Start Your Viewer Playing!"
                            , "http://slguide.com/index.php?v=" + (string)llGetKey() +
                            "&c=" + (string)XML_channel + "&o=" + (string)llGetOwner()
                            + "&");
429                 return;
430             }
431 
432             string header = "Video mode"+moviename+": ";
433 
434             if(message == "<< Prev")
435             {
436                 notecardline--;
437                 if(notecardline < 0)
438                     notecardline = numberofnotecardlines - 1;
439                 tempuser = id;
440                 llGetNotecardLine(NOTECARD, notecardline);
441                 return;
442             }
443             if(message == "Next >>")
```

```
444             {
445                 notecardline++;
446                 if(notecardline >= numberofnotecardlines)
447                     notecardline = 0;
448                 tempuser = id;
449                 llGetNotecardLine(NOTECARD, notecardline);
450                 return;
451             }
452             if(message == "Use")
453             {
454                 if(tempurl == "** No URL specified! **")
455                     tempurl = "";
456                 seturl(tempurl,id);
457                 return;
458             }
459
460             if(message == "Menu")
461             {
462                 menu(id);
463                 return;
464             }
465             if(message == "Configure")
466             {
467                 config(id);
468                 return;
469             }
470             if(message == "Bookmarks")
471             {
472                 if(notecardcheck != -1)
473                 {
474                     llDialog(id,"Error: No valid bookmark data found in notecard '"
                            +NOTECARD+"'.",["Menu"],chan);
475                     return;
476                 }
477                 if(finditem(NOTECARD) != -1)
478                 {
479                     tempuser = id;
480                     if(numberofnotecardlines < notecardline)
481                         notecardline = 0;
482                     llGetNotecardLine(NOTECARD, notecardline);
483                 }
484                 else
485                     llDialog(id,"Error: No notecard named "+NOTECARD+" found in
                            contents.",["Menu"],chan);
486                 return;
487             }
488
489             if(llGetLandOwnerAt(llGetPos()) != llGetOwner())     //If we do not have
                    permissions to actually do the following functions
490             {
491                 llSay(0,"Error: Cannot modify parcel media settings. "+
                        llGetObjectName()+" is not owned by parcel owner.");
492                 menu(id);
493                 return; //Abort
494             }
495
```

```
496             if(listenUrl != -1 && channel == 1) //Incoming data from "Set URL"
                    command (user spoke on channel 1)
497             {
498                 llListenRemove(listenUrl);
499                 listenUrl = -1;
500                 seturl(message,id);
501             }
502             if(message == "Play")
503             {
504                 report(header+"Playing");
505                 llParcelMediaCommandList([PARCEL_MEDIA_COMMAND_PLAY]);
506                 return;
507             }
508             if(message == "Stop")
509             {
510                 report(header+"Stopped");
511                 llParcelMediaCommandList([PARCEL_MEDIA_COMMAND_STOP]);
512                 return;
513             }
514             if(message == "Pause")
515             {
516                 report(header+"Paused");
517                 llParcelMediaCommandList([PARCEL_MEDIA_COMMAND_PAUSE]);
518                 return;
519             }
520             if(message == "Unload")
521             {
522                 report(header+"Stopped");
523                 llParcelMediaCommandList([PARCEL_MEDIA_COMMAND_UNLOAD]);
524                 return;
525             }
526             if(message == "Loop")
527             {
528                 llParcelMediaCommandList([PARCEL_MEDIA_COMMAND_LOOP]);
529                 return;
530             }
531             //URL , Auto-Scale ,
532             if(message == "Set URL")
533             {
534                 report(header+"Stopped");
535                 listenUrl = llListen(1,"",id,"");
536                 llDialog(id,"Please type the URL of your choice with /1 in
                        thebegining. For example, /1 www.google.com",["Ok"],938);
537                 return;
538             }
539             if(message == "Align ON")
540             {
541                 report(header+"Stopped");
542                 llParcelMediaCommandList([PARCEL_MEDIA_COMMAND_AUTO_ALIGN,TRUE]);
543                 menu(id);
544                 return;
545             }
546             if(message == "Align OFF")
547             {
548                 report(header+"Stopped");
549                 llParcelMediaCommandList([PARCEL_MEDIA_COMMAND_AUTO_ALIGN,FALSE]);
550                 menu(id);
```

```
551                 return;
552             }
553             if(message == "Set Remote")
554             {
555                 llSay(0,"Configuring remote...");
556                 encryption = 0;
557                 llListenRemove(listenRemote);
558                 listenRemote = llListen(REMOTE_CHANNEL,"","","");
559                 llSay(REMOTE_CHANNEL,"SETUP");
560             }
561         }
562     }
563
564     dataserver(key queryid, string data)
565     {
566         if(queryid == groupcheck)        //Test if object is deeded to group
567         {
568             groupcheck = NULL_KEY;
569             isGroup = FALSE;
570             return;
571         }
572
573         if(queryid == tempuser) //If just checking number of notecard lines
574         {
575             numberofnotecardlines = (integer)data;
576             notecardkey = llGetInventoryKey(NOTECARD);
577             notecardcheck = 0;
578             llGetNotecardLine(NOTECARD, notecardcheck);
579             return;
580         }
581         if(notecardcheck != -1)
582         {
583             if(data != EOF)
584             {
585                 if(data == "")
586                 {
587                     notecardcheck++;
588                     llGetNotecardLine(NOTECARD, notecardcheck);
589                 }
590                 else
591                 {
592                     notecardcheck = -1;
593                     return;
594                 }
595             }
596             else
597                 return;
598         }
599
600         if(data == "" && notecardline < numberofnotecardlines)     //If user just
                pressed "enter" in bookmarks, skip
601         {
602             notecardline++;
603             llGetNotecardLine(NOTECARD, notecardline);
604             return;
605         }
606
```

```
607         if(data == EOF)
608         {
609             notecardline = 0;
610             llGetNotecardLine(NOTECARD, notecardline);
611             return;
612         }
613         list parsed = llParseString2List(data,["|","| "," |"," | "],[]);     //
                Ensure no blank spaces before "http://".
614         string name = llList2String(parsed,0);
615         tempurl = llList2String(parsed,1);
616         if(tempurl == "")
617             tempurl = "** No URL specified! **";
618
619         tempmoviename = name;
620
621         llDialog(tempuser,"Bookmarks notecard ("+(string)(notecardline+1)+"/"+(
                string)numberofnotecardlines+")\n"+name+" ("+mediatype(llList2String(
                llParseString2List(tempurl,["."],[]),-1))+")\n"+tempurl,["<< Prev","Use
                ","Next >>","Menu"],chan);
622     }
623
624     remote_data(integer type, key channel, key message_id, string sender, integer
            ival, string sval)
625     {
626         if (type == REMOTE_DATA_CHANNEL)
627         {
628             XML_channel = channel;
629         }
630         else if(type == REMOTE_DATA_REQUEST)
631         {
632             list media_info = llParseString2List(sval, ["|"], []);
633             tempmoviename = llList2String(media_info,0);
634             seturl(llList2String(media_info,1),NULL_KEY);
635             llRemoteDataReply(channel, message_id, sval, 1);
636         }
637     }
638
639     timer()
640     {
641         if(llGetTime() > listenTimer)        //If listener time expired...
642         {
643             llListenRemove(listenHandle);    //Remove listeneres.
644             llListenRemove(listenUrl);
645             llListenRemove(listenRemote);
646             listenHandle = -1;
647             listenUrl = -1;
648             listenRemote = -1;
649             listenTimer = -1;
650             if(loop_image == FALSE || mode != 1) //If we're not looping pictures or
                    are in picture mode at all
651                 llSetTimerEvent(0.0);    //Remove timer
652         }
653
654         if(loop_image == TRUE && mode == 1) //If we're looping pictures and and we'
                re in picture mode...
655             next(); //Next picture
656     }
```

```
657 }
658
659 //Retrieved from
660 //   "http://www.simteach.com/wiki/index.php?title=SL_FreeView" ViewsArticle
```

Listing A.6: FreeView 1.2 LSL

A.7 Audio Playlist note card

```
/1|http://secondlife5.inescporto.pt:8000/rage|Rage Against the Machine
/2|http://secondlife5.inescporto.pt:8000/deepdish|Deepdish - Say Hello
/3|http://secondlife5.inescporto.pt:8000/south|Musics from South Park
/4|http://secondlife5.inescporto.pt:8000/poetryofshadow|Poetry of Shadows - Spheres
    of Knowledge
/5|http://secondlife5.inescporto.pt:8000/remixescd1|Remixes 07 cd1
/6|http://secondlife5.inescporto.pt:8000/remixescd2|Remixes 07 cd2
/7|http://secondlife5.inescporto.pt:8000/matrix|Matrix OST
/8|http://secondlife5.inescporto.pt:8000/tatu| t.a.t.u.+Ramstein
/9|http://secondlife5.inescporto.pt:8000/baladadadespedida|Balada Da Despedida Do 6
    o Ano Medico 1958
/10|http://secondlife5.inescporto.pt:8000/matisyahu|Matisyahu - Jerusalem
/11|http://secondlife5.inescporto.pt:8000/anuk|Anuk - You Can Make Me Whole Again
/12|http://secondlife5.inescporto.pt:8000/britney|Britney Spears - Breathe On Me
/13|http://secondlife5.inescporto.pt:8000/zoomzoomzoom|CAPOEIRA SONG - ZOOM ZOOM
    ZOOM (SOUNDTRACK ONLY THE STRONG)
/14|http://secondlife5.inescporto.pt:8000/telemovel|Da-me O Telemovel Ja! (Remix)
/15|http://secondlife5.inescporto.pt:8000/djkool|dj kool - let me clear my throat
/16|http://secondlife5.inescporto.pt:8000/editors|Editors
/17|http://secondlife5.inescporto.pt:8000/fcp|FCPorto - Hino do Futebol Clube do
    Porto
```

Listing A.7: Audio Playlist note card

A.8 Video Playlist note card

```
/1|rtsp://secondlife5.inescporto.pt/sample_h264_1mbit.mp4
/2|rtsp://secondlife5.inescporto.pt/sample_h264_300kbit.mp4
/3|rtsp://secondlife5.inescporto.pt/fun.sdp
/4|rtsp://secondlife5.inescporto.pt/test.sdp
/5|rtsp://secondlife5.inescporto.pt/9e8d6b12df7a2e38.mp4
/6|rtsp://secondlife5.inescporto.pt/MP4.sdp
/7|rtsp://secondlife5.inescporto.pt/mov.sdp
/8|rtsp://secondlife5.inescporto.pt/20080511-f-f-f-f-f.mp4
/9|rtsp://secondlife5.inescporto.pt/20080512-human-robots.mp4
/10|rtsp://secondlife5.inescporto.pt/20080513-sea-office.mp4
/11|rtsp://secondlife5.inescporto.pt/20080514-funny-maths.mp4
/12|rtsp://secondlife5.inescporto.pt/20080515-musical-food-court.mp4
/13|rtsp://secondlife5.inescporto.pt/20080516-happy-mothers-day-milfs.mp4
/14|rtsp://secondlife5.inescporto.pt/20080517-cool-videography.mp4
/15|rtsp://secondlife5.inescporto.pt/20080518-self-service-toilet-bowl.mp4
/16|rtsp://secondlife5.inescporto.pt/20080519-stout-with-love.mp4
/17|rtsp://secondlife5.inescporto.pt/20080520-crazy-human-snake.mp4
/18|rtsp://secondlife5.inescporto.pt/20080521-beer-trap.mp4
/19|rtsp://secondlife5.inescporto.pt/20080522-face-and-hand-art.mp4
/20|rtsp://secondlife5.inescporto.pt/9e8d6b12df7a2e38.mp4
/21|rtsp://secondlife5.inescporto.pt/ironman-lo.mov
/22|rtsp://secondlife5.inescporto.pt/missreef.mp4
/23|rtsp://secondlife5.inescporto.pt/NotaMissReef2007.mp4
/24|rtsp://secondlife5.inescporto.pt/TheVeronicas_HookMeUp.mp4
/25|rtsp://secondlife5.inescporto.pt/[OPFansMaplesnow][one_piece][350][jap_chn][
    HDTV][x264_aac][1920x1080][crc_604D3255].mp4
```

Listing A.8: Video Playlist note card

For more information regarding LSL visit the Second Life wiki[55].

Appendix B

Java code

B.1 iPlay

```
1  package gui;
2
3  import server.SLserver;
4  import server.StateObj;
5  import filesdb.Playlist;
6  import filesdb.Userlist;
7  import javax.swing.JPanel;
8  import java.awt.GraphicsConfiguration;
9  import java.awt.HeadlessException;
10 import java.io.IOException;
11 import javax.swing.JFrame;
12 import javax.swing.JButton;
13 import javax.swing.JCheckBox;
14 import javax.swing.JLabel;
15 import javax.swing.ImageIcon;
16 import client.SLClient;
17 import com.phidgets.PhidgetException;
18 import com.phidgets.RFIDPhidget;
19 import com.phidgets.event.AttachEvent;
20 import com.phidgets.event.AttachListener;
21 import com.phidgets.event.DetachEvent;
22 import com.phidgets.event.DetachListener;
23 import com.phidgets.event.ErrorEvent;
24 import com.phidgets.event.ErrorListener;
25 import com.phidgets.event.OutputChangeEvent;
26 import com.phidgets.event.OutputChangeListener;
27 import com.phidgets.event.TagGainEvent;
28 import com.phidgets.event.TagGainListener;
29 import com.phidgets.event.TagLossEvent;
30 import com.phidgets.event.TagLossListener;
31 import javax.swing.JTextField;
32 import javax.swing.JToggleButton;
33 import javax.swing.JComboBox;
34
35
36
37 public class IPlay extends JFrame {
```

```
38
39    private static final long serialVersionUID = 1L;
40    private static Thread server;
41    private JPanel jContentPane = null;
42    private JButton jButtonFF = null;
43    private JButton jButtonPlay = null;
44    private JButton jButtonFB = null;
45    private int os=0;
46    private String [] playerpath=null;
47    private int windows=0;
48    private int unix=1;
49    private int mac=2;
50    private int presentState=0;
51    SLClient slc = null;
52    Userlist users =null;
53    Playlist playlist=null;
54    private JCheckBox jCheckBoxUser1 = null;
55    private JCheckBox jCheckBoxUser2 = null;
56    private JLabel jLabelPrevious = null;
57    private JLabel jLabelPlaying = null;
58    private JLabel jLabelNext = null;
59    private JLabel jLabelback = null;
60    private JLabel jLabelState = null;
61    RFIDPhidget rfid;
62    private JTextField jTextTag1 = null;
63    private JTextField jTextTag2 = null;
64    private JButton jButtonAddUser = null;
65    private JButton jButtonAddSong = null;
66    private JFrame jFrameSong = null;
67    private JPanel jContentPaneS = null;
68    private JFrame jFrameAddUser = null;
69    private JPanel jContentPaneU = null;
70    private JButton jButtonAddUserOk = null;
71    private JTextField jTextFieldName = null;
72    private JTextField jTextFieldTag = null;
73    private JTextField jTextFieldAvatar = null;
74    private JTextField jTextFieldUID = null;
75    private JLabel jLabelAU1 = null;
76    private JLabel jLabel = null;
77    private JLabel jLabel1 = null;
78    private JLabel jLabel2 = null;
79    private StateObj finalState=null;
80    private String [] sys={"Windos","Linux","MacOSX"};
81    private String url=null;
82    private String musicName=null;
83    private int playingIndex=0;
84    private boolean SLon = false ;
85    private boolean RFIDon = false ;
86    private JToggleButton jToggleLink = null;
87    private JToggleButton jToggleRFID = null;
88    private JComboBox jComboBoxSystem = null;
89    private boolean playRemoteFlag;
90
91
92    public IPlay(StateObj state) throws HeadlessException {
93      super();
94      this.finalState=state;
```

```
95      initialize();
96      (this.finalState.getJCheckBoxState()).addItemListener(new java.awt.event.
            ItemListener() {
97        public void itemStateChanged(java.awt.event.ItemEvent e) {
98          System.out.println("\n\n\n\n!!!!!!!!!!!!!!!!!!!Remote Play
                !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!\n\n\n");
99          playingIndex=Integer.parseInt(finalState.get());
100         playRemote();
101         System.out.println("\n\n\n\n!!!!!!!!!!!!!!!!!!!After play
                !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!\n\n\n");
102         return;
103       }
104     });
105   }
106
107   public void setOS(int osx){
108     this.os=osx;
109
110   }
111
112   public int getOS(){
113
114     return this.os;
115   }
116
117   public IPlay(GraphicsConfiguration arg0) {
118     super(arg0);
119     initialize();
120   }
121
122   public IPlay(String arg0) throws HeadlessException {
123     super(arg0);
124     initialize();
125   }
126
127   public IPlay(String arg0, GraphicsConfiguration arg1) {
128     super(arg0, arg1);
129     initialize();
130   }
131
132   private JButton getJButtonFF() {
133     if (jButtonFF == null) {
134       jButtonFF = new JButton();
135       jButtonFF.setFont(new java.awt.Font("Comic Sans MS", java.awt.Font.BOLD, 18))
              ;
136       jButtonFF.setBounds(new java.awt.Rectangle(365,168,60,60));
137       jButtonFF.setIcon(new ImageIcon(getClass().getResource("/gui/FF.jpg")));
138       jButtonFF.setText("");
139       jButtonFF.addActionListener(new java.awt.event.ActionListener() {
140         public void actionPerformed(java.awt.event.ActionEvent e) {
141           System.out.println("actionPerformed() FF");
142           FF();
143         }
144       });
145     }
146     return jButtonFF;
147   }
```

```
148
149   private JButton getJButtonPlay() {
150     if (jButtonPlay == null) {
151       jButtonPlay = new JButton();
152       jButtonPlay.setFont(new java.awt.Font("Comic Sans MS", java.awt.Font.BOLD,
              18));
153       jButtonPlay.setBounds(new java.awt.Rectangle(261,168,60,60));
154       jButtonPlay.setIcon(new ImageIcon(getClass().getResource("/gui/Windows Media
              Player play.jpg")));
155       jButtonPlay.setText("");
156       jButtonPlay.addActionListener(new java.awt.event.ActionListener() {
157         public void actionPerformed(java.awt.event.ActionEvent e) {
158           System.out.println("actionPerformed()"); // TODO Auto-generated Event
                  stub actionPerformed()
159           play();
160         }
161       });
162     }
163     return jButtonPlay;
164   }
165
166   private JButton getJButtonFB() {
167     if (jButtonFB == null) {
168       jButtonFB = new JButton();
169       jButtonFB.setBounds(new java.awt.Rectangle(158,168,60,60));
170       jButtonFB.setFont(new java.awt.Font("Comic Sans MS", java.awt.Font.BOLD, 18))
              ;
171       jButtonFB.setIcon(new ImageIcon(getClass().getResource("/gui/FB.jpg")));
172       jButtonFB.setText("");
173       jButtonFB.addActionListener(new java.awt.event.ActionListener() {
174         public void actionPerformed(java.awt.event.ActionEvent e) {
175           System.out.println("Previous"); // TODO Auto-generated Event stub
                  actionPerformed()
176           FB();
177         }
178       });
179     }
180     return jButtonFB;
181   }
182
183   private JCheckBox getJCheckBoxUser1() {
184     if (jCheckBoxUser1 == null) {
185       jCheckBoxUser1 = new JCheckBox();
186       jCheckBoxUser1.setBounds(new java.awt.Rectangle(0,0,113,20));
187       jCheckBoxUser1.setBackground(java.awt.Color.white);
188       jCheckBoxUser1.setText("Tag1 detected");
189       jCheckBoxUser1.addItemListener(new java.awt.event.ItemListener() {
190         public void itemStateChanged(java.awt.event.ItemEvent e) {
191           System.out.println("itemStateChanged()");
192           if(jCheckBoxUser2.isSelected()){
193             if(jCheckBoxUser1.isSelected())
194               presentState=3;//both tags are present
195             else
196               presentState=2;//only tag 2 is present
197           }
198           else{
199             if(jCheckBoxUser1.isSelected())
```

```
200                 presentState=1;//only tag1 is present
201               else
202                 presentState=0;// nothing
203             }
204             sendSensorData(presentState);
205           }
206         });
207     }
208     return jCheckBoxUser1;
209   }
210
211   private JCheckBox getJCheckBoxUser2() {
212     if (jCheckBoxUser2 == null) {
213       jCheckBoxUser2 = new JCheckBox();
214       jCheckBoxUser2.setBounds(new java.awt.Rectangle(112,0,119,20));
215       jCheckBoxUser2.setBackground(java.awt.Color.white);
216       //jCheckBoxUser2.setEnabled(false);
217       jCheckBoxUser2.setText("Tag2 detected");
218       jCheckBoxUser2.addItemListener(new java.awt.event.ItemListener() {
219         public void itemStateChanged(java.awt.event.ItemEvent e) {
220           System.out.println("itemStateChanged() checkbox2");
221           if(jCheckBoxUser1.isSelected()){
222             if(jCheckBoxUser2.isSelected())
223               presentState=3;//both tags are present
224             else
225               presentState=1;//only tag 1 is present
226           }
227           else{
228             if(jCheckBoxUser2.isSelected())
229               presentState=2;//only tag2 is present
230             else
231               presentState=0;// nothing
232           }
233           sendSensorData(presentState);
234           System.out.println("presentState: "+presentState);
235         }
236       });
237
238     }
239     return jCheckBoxUser2;
240   }
241
242   private JTextField getJTextTag1() {
243     if (jTextTag1 == null) {
244       jTextTag1 = new JTextField();
245       jTextTag1.setBounds(new java.awt.Rectangle(5,20,103,20));
246       jTextTag1.setEditable(false);
247     }
248     return jTextTag1;
249   }
250
251   private JTextField getJTextTag2() {
252     if (jTextTag2 == null) {
253       jTextTag2 = new JTextField();
254       jTextTag2.setBounds(new java.awt.Rectangle(117,20,103,20));
255       jTextTag2.setEditable(false);
256     }
```

```
257     return jTextTag2;
258   }
259
260   private JButton getJButtonAddUser() {
261     if (jButtonAddUser == null) {
262       jButtonAddUser = new JButton();
263       jButtonAddUser.setBounds(new java.awt.Rectangle(363,23,96,20));
264       jButtonAddUser.setFont(new java.awt.Font("Dialog", java.awt.Font.PLAIN, 10));
265       jButtonAddUser.setText("Add User");
266       jButtonAddUser.addActionListener(new java.awt.event.ActionListener() {
267         public void actionPerformed(java.awt.event.ActionEvent e) {
268           System.out.println("actionPerformed()"); // TODO Auto-generated Event
                stub actionPerformed()
269           getJFrameAddUser();
270           jFrameAddUser.setVisible(true);
271           jFrameAddUser.setEnabled(true);
272         }
273       });
274
275
276
277     }
278     return jButtonAddUser;
279   }
280
281   private JButton getJButtonAddSong() {
282     if (jButtonAddSong == null) {
283       jButtonAddSong = new JButton();
284       jButtonAddSong.setBounds(new java.awt.Rectangle(363,2,96,20));
285       jButtonAddSong.setFont(new java.awt.Font("Dialog", java.awt.Font.PLAIN, 10));
286       jButtonAddSong.setText("Add Song");
287       jButtonAddSong.addActionListener(new java.awt.event.ActionListener() {
288         public void actionPerformed(java.awt.event.ActionEvent e) {
289           System.out.println("actionPerformed()"); // TODO Auto-generated Event
                stub actionPerformed()
290           getJFrameSong();
291           jFrameSong.setVisible(true);
292           jFrameSong.setEnabled(true);
293         }
294       });
295     }
296     return jButtonAddSong;
297   }
298
299   private JFrame getJFrameSong() {
300     if (jFrameSong == null) {
301       jFrameSong = new JFrame();
302       jFrameSong.setSize(new java.awt.Dimension(372,226));
303       jFrameSong.setTitle("iPlay|Add Song");
304       jFrameSong.setContentPane(getJContentPaneS());
305     }
306     return jFrameSong;
307   }
308
309   private JPanel getJContentPaneS() {
310     if (jContentPaneS == null) {
311       jContentPaneS = new JPanel();
```

```
312         jContentPaneS.setLayout(null);
313       }
314       return jContentPaneS;
315     }
316
317     private JFrame getJFrameAddUser() {
318       if (jFrameAddUser == null) {
319         jFrameAddUser = new JFrame();
320         jFrameAddUser.setTitle("iPlay|Add User");
321         jFrameAddUser.setBounds(new java.awt.Rectangle(250,250,372,220));
322         jFrameAddUser.setEnabled(false);
323         jFrameAddUser.setVisible(false);
324         jFrameAddUser.setContentPane(getJContentPaneU());
325       }
326       return jFrameAddUser;
327     }
328
329     private JPanel getJContentPaneU() {
330       if (jContentPaneU == null) {
331         jLabel2 = new JLabel();
332         jLabel2.setBounds(new java.awt.Rectangle(17,100,80,25));
333         jLabel2.setText("UID:");
334         jLabel1 = new JLabel();
335         jLabel1.setBounds(new java.awt.Rectangle(17,70,80,25));
336         jLabel1.setText("Avatar:");
337         jLabel = new JLabel();
338         jLabel.setBounds(new java.awt.Rectangle(17,40,80,25));
339         jLabel.setText("Tag:");
340         jLabelAU1 = new JLabel();
341         jLabelAU1.setBounds(new java.awt.Rectangle(16,10,80,25));
342         jLabelAU1.setVerticalAlignment(javax.swing.SwingConstants.CENTER);
343         jLabelAU1.setText("Name:");
344         jContentPaneU = new JPanel();
345         jContentPaneU.setLayout(null);
346         jContentPaneU.add(getJButtonAddUserOk(), null);
347         jContentPaneU.add(getJTextFieldName(), null);
348         jContentPaneU.add(getJTextFieldTag(), null);
349         jContentPaneU.add(getJTextFieldAvatar(), null);
350         jContentPaneU.add(getJTextFieldUID(), null);
351         jContentPaneU.add(jLabelAU1, null);
352         jContentPaneU.add(jLabel, null);
353         jContentPaneU.add(jLabel1, null);
354         jContentPaneU.add(jLabel2, null);
355       }
356       return jContentPaneU;
357     }
358
359     private JButton getJButtonAddUserOk() {
360       if (jButtonAddUserOk == null) {
361         jButtonAddUserOk = new JButton();
362         jButtonAddUserOk.setBounds(new java.awt.Rectangle(15,152,330,28));
363         jButtonAddUserOk.setText("ADD");
364         jButtonAddUserOk.addActionListener(new java.awt.event.ActionListener() {
365           public void actionPerformed(java.awt.event.ActionEvent e) {
366             System.out.println("actionPerformed()"); // TODO Auto-generated Event
                  stub actionPerformed()
367
```

```
368             String [] data = new String[5];
369             data[0]=jTextFieldName.getText();
370             data[1]=jTextFieldTag.getText();
371             data[2]=jTextFieldAvatar.getText();
372             data[3]=jTextFieldUID.getText();
373             data[4]=jTextFieldName.getText();
374 
375             try {
376               users.saveUserlist(data);
377             } catch (IOException e1) {
378               // TODO Auto-generated catch block
379               e1.printStackTrace();
380             }
381           }
382         });
383     }
384     return jButtonAddUserOk;
385   }
386 
387   private JTextField getJTextFieldName() {
388     if (jTextFieldName == null) {
389       jTextFieldName = new JTextField();
390       jTextFieldName.setBounds(new java.awt.Rectangle(104,10,226,25));
391     }
392     return jTextFieldName;
393   }
394 
395   private JTextField getJTextFieldTag() {
396     if (jTextFieldTag == null) {
397       jTextFieldTag = new JTextField();
398       jTextFieldTag.setBounds(new java.awt.Rectangle(104,40,226,25));
399     }
400     return jTextFieldTag;
401   }
402 
403   private JTextField getJTextFieldAvatar() {
404     if (jTextFieldAvatar == null) {
405       jTextFieldAvatar = new JTextField();
406       jTextFieldAvatar.setBounds(new java.awt.Rectangle(104,70,226,25));
407     }
408     return jTextFieldAvatar;
409   }
410 
411   private JTextField getJTextFieldUID() {
412     if (jTextFieldUID == null) {
413       jTextFieldUID = new JTextField();
414       jTextFieldUID.setBounds(new java.awt.Rectangle(104,100,226,25));
415     }
416     return jTextFieldUID;
417   }
418 
419   private JToggleButton getJToggleLink() {
420     if (jToggleLink == null) {
421       jToggleLink = new JToggleButton();
422       jToggleLink.setBounds(new java.awt.Rectangle(459,2,80,20));
423       jToggleLink.setEnabled(true);
424       jToggleLink.setText("SL");
```

```
425       jToggleLink.addItemListener(new java.awt.event.ItemListener() {
426         public void itemStateChanged(java.awt.event.ItemEvent e) {
427           System.out.println("itemStateChanged() (SL online)"); // TODO Auto-
                generated Event stub itemStateChanged()
428           if(SLon)
429             SLon=false;
430           else
431             SLon=true;
432           System.out.println("SL online: "+SLon);
433         }
434       });
435     }
436     return jToggleLink;
437   }
438
439   private JToggleButton getJToggleRFID() {
440     if (jToggleRFID == null) {
441       jToggleRFID = new JToggleButton();
442       jToggleRFID.setBounds(new java.awt.Rectangle(459,23,80,20));
443       jToggleRFID.setText("RFID");
444       jToggleRFID.addItemListener(new java.awt.event.ItemListener() {
445         public void itemStateChanged(java.awt.event.ItemEvent e) {
446           System.out.println("itemStateChanged() (RFID on)"); // TODO Auto-
                generated Event stub itemStateChanged()
447           if(RFIDon)
448             RFIDon=false;
449           else
450             RFIDon=true;
451           System.out.println("RFID on: "+RFIDon);
452           jCheckBoxUser1.setEnabled(!RFIDon);
453           jCheckBoxUser2.setEnabled(!RFIDon);
454           if(RFIDon)
455             initPhidget();
456           else
457             closePhidget();
458         }
459       });
460     }
461     return jToggleRFID;
462   }
463
464   private JComboBox getJComboBoxSystem() {
465     if (jComboBoxSystem == null) {
466       jComboBoxSystem = new JComboBox(this.sys);
467       jComboBoxSystem.setBounds(new java.awt.Rectangle(538,2,100,20));
468       jComboBoxSystem.addItemListener(new java.awt.event.ItemListener() {
469         public void itemStateChanged(java.awt.event.ItemEvent e) {
470           System.out.println("itemStateChanged()"); // TODO Auto-generated Event
                stub itemStateChanged()
471           setOS(jComboBoxSystem.getSelectedIndex());
472           System.out.println("OS:"+getOS()+"");
473         }
474       });
475     }
476     return jComboBoxSystem;
477   }
478
```

```
479   public static void main(String[] args){
480
481         final StateObj State=new StateObj();
482         (new Thread(new Runnable(){
483           public void run(){
484             new IPlay(State).setVisible(true);
485           }
486         })).start();
487         (server= new Thread(new SLserver(State))).start();
488
489   }
490
491
492   private void initialize() {
493     slc=new SLClient();
494     users= new Userlist();
495     playlist= new Playlist();
496     this.url=(this.playlist.getSongs())[this.playingIndex].getURL();
497     this.playerpath=new String[3];
498     this.playerpath[windows]="\"C:\\Program Files\\Windows Media Player\\wmplayer.
          exe\" /fullscreen ";
499     this.playerpath[unix]="firefox";
500     this.playerpath[mac]="/Applications/Firefox.app/Contents/MacOS/firefox";
501     if(RFIDon)
502       initPhidget();
503     this.setSize(656, 263);
504     this.setContentPane(getJContentPane());
505     this.setTitle("iPlay");
506     this.addWindowListener(new java.awt.event.WindowAdapter() {
507       public void windowClosing(java.awt.event.WindowEvent e) {
508         System.out.println("windowClosing()");
509         if(RFIDon)
510           closePhidget();
511         try {
512           this.finalize();
513           server.join(1000);
514           server.interrupt();
515         } catch (Throwable e1) {
516           e1.printStackTrace();
517         }
518       }
519     });
520   }
521
522   private JPanel getJContentPane() {
523     if (jContentPane == null) {
524       jLabelState = new JLabel();
525       jLabelState.setBounds(new java.awt.Rectangle(230,8,115,26));
526       jLabelState.setDisabledIcon(new ImageIcon(getClass().getResource("/gui/label.
            jpg")));
527       jLabelState.setText("");
528       jLabelback = new JLabel();
529       jLabelback.setBounds(new java.awt.Rectangle(0,46,660,122));
530       jLabelback.setIcon(new ImageIcon(getClass().getResource("/gui/label.jpg")));
531       jLabelback.setText("");
532       jLabelNext = new JLabel();
533       jLabelNext.setBounds(new java.awt.Rectangle(424,176,158,40));
```

```
534         jLabelNext.setText((this.playlist.getSongs())[(this.playingIndex+1)].
                getSongName());
535         jLabelPlaying = new JLabel();
536         jLabelPlaying.setBounds(new java.awt.Rectangle(212,128,158,40));
537         jLabelPlaying.setText((this.playlist.getSongs())[this.playingIndex].
                getSongName());
538         jLabelPrevious = new JLabel();
539         jLabelPrevious.setBounds(new java.awt.Rectangle(0,175,158,40));
540         jLabelPrevious.setText((this.playlist.getSongs())[(this.playlist.
                getNmuberOfSongs()-1)].getSongName());
541         jContentPane = new JPanel();
542         jContentPane.setLayout(null);
543         jContentPane.setBackground(java.awt.Color.white);
544         jContentPane.add(getJButtonFF(), null);
545         jContentPane.add(getJButtonPlay(), null);
546         jContentPane.add(getJButtonFB(), null);
547         jContentPane.add(getJCheckBoxUser1(), null);
548         jContentPane.add(getJCheckBoxUser2(), null);
549         jContentPane.add(jLabelPrevious, null);
550         jContentPane.add(jLabelPlaying, null);
551         jContentPane.add(jLabelNext, null);
552         jContentPane.add(jLabelback, null);
553         jContentPane.add(jLabelState, null);
554         jContentPane.add(getJTextTag1(), null);
555         jContentPane.add(getJTextTag2(), null);
556         jContentPane.add(getJButtonAddUser(), null);
557         jContentPane.add(getJButtonAddSong(), null);
558         jContentPane.add(getJToggleLink(), null);
559         jContentPane.add(getJToggleRFID(), null);
560         jContentPane.add(getJComboBoxSystem(), null);
561       }
562       return jContentPane;
563     }
564
565     public void sendSensorData(int state){
566       String tempURL="http://www.nothing.pt",tempSongName="";
567       switch (state) {
568               case 0:  break;
569               case 1:  tempURL=(this.playlist.getSongs())[0].getURL();
570                        tempSongName=(this.playlist.getSongs())[0].getSongName();
571                        break;
572               case 2:  tempURL=(this.playlist.getSongs())[1].getURL();
573                tempSongName=(this.playlist.getSongs())[1].getSongName();
574                break;
575               case 3:  tempURL=(this.playlist.getSongs())[2].getURL();
576                tempSongName=(this.playlist.getSongs())[2].getSongName();
577                break;
578               default: System.out.println("Invalid State.");break;
579       }
580         if(SLon){
581           slc.send("sensor|"+state+"|"+tempURL+"|"+tempSongName);
582         }
583       if(getOS()==windows){
584       String comando = playerpath[windows]+"\""+tempURL+"\"";
585       System.out.println("Comando: "+comando+"");
586       try {
```

```
587           Runtime.getRuntime().exec("rundll32 SHELL32.DLL,ShellExec_RunDLL "+comando)
                  ;
588         } catch (IOException e1) {
589           e1.printStackTrace();
590         }
591       }
592       else if(getOS()==unix){
593         String comando =""+playerpath[unix]+" "+tempURL+"";
594         try {
595           Runtime.getRuntime().exec(comando);
596         } catch (IOException e1) {
597           e1.printStackTrace();
598         }
599       }
600       else if(getOS()==mac){
601         String comando =playerpath[mac]+" "+url;
602         System.out.println("Comando: "+comando+"");
603         try {
604           Runtime.getRuntime().exec(comando);
605         } catch (IOException e1) {
606           e1.printStackTrace();
607         }
608       }
609       else System.out.println("Choose OS!!!");
610     }
611
612     public void initPhidget(){
613         try {
614           rfid = new RFIDPhidget();
615           rfid.openAny();
616           System.out.println("waiting for RFID attachment...");
617           rfid.waitForAttachment(1000);
618           System.out.println("Serial: " + rfid.getSerialNumber());
619           System.out.println("Outputs: " + rfid.getOutputCount());
620           rfid.setAntennaOn(true);
621           rfid.setLEDOn(true);
622
623         } catch (PhidgetException e1) {
624           e1.printStackTrace();
625         }
626         rfid.addAttachListener(new AttachListener() {
627           public void attached(AttachEvent ae)
628           {
629             try
630             {
631               ((RFIDPhidget)ae.getSource()).setAntennaOn(true);
632               ((RFIDPhidget)ae.getSource()).setLEDOn(true);
633             }
634             catch (PhidgetException ex) { }
635             System.out.println("attachment of " + ae);
636           }
637         });
638         rfid.addDetachListener(new DetachListener() {
639           public void detached(DetachEvent ae) {
640             System.out.println("detachment of " + ae);
641           }
642         });
```

```
643         rfid.addErrorListener(new ErrorListener() {
644           public void error(ErrorEvent ee) {
645             System.out.println("error event for " + ee);
646           }
647         });
648         rfid.addTagGainListener(new TagGainListener()
649         {
650           public void tagGained(TagGainEvent oe)
651           {
652             System.out.println(oe);
653             String userTag = (oe.getValue().split(" ", 6))[0];
654             if(userTag.equals("01069351b3")){
655               jCheckBoxUser1.setSelected(true);
656               jTextTag1.setText(userTag);
657             }
658             else if(userTag.equals("0106934345")){
659               jCheckBoxUser2.setSelected(true);
660               jTextTag2.setText(userTag);
661             }
662           }
663         });
664         rfid.addTagLossListener(new TagLossListener()
665         {
666           public void tagLost(TagLossEvent oe)
667           {
668             System.out.println(oe);
669             String userTag = (oe.getValue().split(" ", 6))[0];
670             if(userTag.equals("01069351b3")){
671               jCheckBoxUser1.setSelected(false);
672               jTextTag1.setText("");
673             }
674             else if(userTag.equals("0106934345")){
675               jCheckBoxUser2.setSelected(false);
676               jTextTag2.setText("");
677             }
678           }
679         });
680         rfid.addOutputChangeListener(new OutputChangeListener()
681         {
682           public void outputChanged(OutputChangeEvent oe)
683           {
684             System.out.println(oe);
685           }
686         });
687     }
688
689     public void closePhidget(){
690
691         try
692         {
693           rfid.setAntennaOn(false);
694           rfid.setLEDOn(false);
695         }
696         catch (PhidgetException ex) { System.out.println("attachment of " + ex);}
697
698         try {
699
```

```
700         rfid.close();
701       } catch (PhidgetException e1) {
702         e1.printStackTrace();
703       } catch (Throwable ef) {
704         ef.printStackTrace();
705       }
706       rfid = null;
707   }
708
709   public void playRemote(){
710     playRemoteFlag=true;
711     if(this.playingIndex==0){//it came from SL
712       musicName="";
713       url="http://";//if you put "" it still plays whatever was playing
714       play();
715       System.out.println("URL(test): "+url);
716       playRemoteFlag=false;
717       return;
718     }
719     int indextmp=(playingIndex-1);
720     url=(this.playlist.getSongs())[indextmp].getURL();
721     musicName=(this.playlist.getSongs())[indextmp].getSongName();
722     play();
723     playRemoteFlag=false;
724   }
725
726   public void play(){
727
728       if(SLon&&(!playRemoteFlag)){
729         musicName=(this.playlist.getSongs())[this.playingIndex].getSongName();
730         slc.send("playURL|"+url+"|"+musicName);
731       }
732       System.out.println("Now Playing "+musicName+"!");
733       if(getOS()==windows){
734         String comando = playerpath[windows]+"\""+url+"\"";
735         System.out.println("Comando: "+comando+"");
736         try {
737           Runtime.getRuntime().exec("rundll32 SHELL32.DLL,ShellExec_RunDLL "+
                comando);
738         } catch (IOException e1) {
739           // TODO Auto-generated catch block
740           e1.printStackTrace();
741         }
742       }
743       else if(getOS()==unix){
744         String comando =""+playerpath[unix]+" "+url+"";
745         try {
746           Runtime.getRuntime().exec(comando);
747         } catch (IOException e1) {
748           // TODO Auto-generated catch block
749           e1.printStackTrace();
750         }
751       }
752       else if(getOS()==mac){
753         String comando =playerpath[mac]+" "+url;
754         System.out.println("Comando: "+comando+"");
755         try {
```

```
756             Runtime.getRuntime().exec(comando);
757           } catch (IOException e1) {
758             // TODO Auto-generated catch block
759             e1.printStackTrace();
760           }
761         }
762         else System.out.println("Choose OS!!!");
763     }
764
765
766
767     public void FF(){
768       int tmpPreviousIndex=0,tmpNextIndex=0;
769       if(this.playingIndex<(this.playlist.getNmuberOfSongs()-1)){
770         tmpPreviousIndex=this.playingIndex;
771         this.playingIndex++;
772         tmpNextIndex=(this.playingIndex+1);
773         if(this.playingIndex==(this.playlist.getNmuberOfSongs()-1))
774           tmpNextIndex=0;
775       }else{
776         System.out.println("ELSE...");
777         tmpPreviousIndex=(this.playlist.getNmuberOfSongs()-1);
778         this.playingIndex=0;
779         tmpNextIndex=1;
780       }
781
782       jLabelPrevious.setText((this.playlist.getSongs())[tmpPreviousIndex].getSongName
              ());
783       jLabelPlaying.setText((this.playlist.getSongs())[this.playingIndex].getSongName
              ());
784       jLabelNext.setText((this.playlist.getSongs())[tmpNextIndex].getSongName());
785       this.url=(this.playlist.getSongs())[this.playingIndex].getURL();
786     }
787
788     public void FB(){
789       int tmpPreviousIndex=0,tmpNextIndex=0;
790       if(this.playingIndex>0){
791         tmpNextIndex=this.playingIndex;
792         this.playingIndex--;
793         tmpPreviousIndex=(this.playingIndex-1);
794         if(this.playingIndex==0)
795           tmpPreviousIndex=(this.playlist.getNmuberOfSongs()-1);
796       }else{
797         System.out.println("ELSE...");
798         tmpPreviousIndex=(this.playlist.getNmuberOfSongs()-2);
799         this.playingIndex=(this.playlist.getNmuberOfSongs()-1);
800         tmpNextIndex=1;
801       }
802       jLabelPrevious.setText((this.playlist.getSongs())[tmpPreviousIndex].getSongName
              ());
803       jLabelPlaying.setText((this.playlist.getSongs())[this.playingIndex].getSongName
              ());
804       jLabelNext.setText((this.playlist.getSongs())[tmpNextIndex].getSongName());
805       this.url=(this.playlist.getSongs())[this.playingIndex].getURL();
806     }
807
808
```

```
809 }
```

Listing B.1: iPlay

B.2 StateObj

```
1  package server;
2  import javax.swing.JCheckBox;
3
4  public class StateObj {
5      private String finalState=null;
6    private JCheckBox jCheckBoxState = null;
7
8      public synchronized String get() {
9          notifyAll();
10         return finalState;
11     }
12
13     public synchronized void set(String state) {
14         this.finalState = state;
15         System.out.println("State obj changed: " +state);
16         notifyAll();
17     }
18
19     public JCheckBox getJCheckBoxState() {
20     if (jCheckBoxState == null) {
21       jCheckBoxState = new JCheckBox();
22     }
23     return jCheckBoxState;
24   }
25 }
```

Listing B.2: StateObj

B.3 SLserver

```
1  package server;
2   /*
3   * Based on EchoServer.java "Accept an HTTP request and echo it back as the HTTP
        response"
4   */
5
6
7  import static java.net.HttpURLConnection.HTTP_OK;
8
9  import java.io.IOException;
10 import java.io.InputStream;
11 import java.io.OutputStream;
12 import java.net.InetSocketAddress;
13 import java.net.URLDecoder;
14 import java.util.List;
15 import java.util.concurrent.Executors;
16
17 import com.sun.net.httpserver.Headers;
18 import com.sun.net.httpserver.HttpExchange;
19 import com.sun.net.httpserver.HttpHandler;
20 import com.sun.net.httpserver.HttpServer;
21
22 public class SLserver implements Runnable{
23
24   private StateObj state;
25
26   public SLserver(StateObj state) {
27     this.state=state;
28   }
29
30   public void stateChange(String newstate) {
31     this.state.set(newstate);
32   }
33
34   public class StateHandler implements HttpHandler {
35     public void handle(HttpExchange t) throws IOException {
36           final InputStream is;
37           final OutputStream os;
38           StringBuilder buf;
39           int b;
40           final String request, response;
41
42           buf = new StringBuilder();
43           System.out.println("Handling Request...");
44           is = t.getRequestBody();
45           System.out.println("Request Body: "+is.toString());
46
47           while ((b = is.read()) != -1) {
48               buf.append((char) b);
49               System.out.println(b);
50           }
51           is.close();
52           System.out.println("\n\n*****REQUEST*****:\n\n");
53           if (buf.length() > 0) {
54               request = URLDecoder.decode(buf.toString(), "UTF-8");
```

```
55              System.out.println(request);
56              state.set(request);
57              if(state.getJCheckBoxState().isSelected())
58                state.getJCheckBoxState().setSelected(false);
59              else
60                state.getJCheckBoxState().setSelected(true);
61
62          } else {
63              request = null;
64          }
65
66          buf = new StringBuilder();
67          buf.append("<html><head><title>HTTP echo server</title></head><body>");
68          buf.append("<p><pre>");
69          buf.append(t.getRequestMethod() + " " + t.getRequestURI() + " " + t.
                getProtocol() + "\n");
70
71          Headers headers = t.getRequestHeaders();
72
73          for (String name : headers.keySet()) {
74              List<String> values = headers.get(name);
75
76              for (String value : values) {
77                  buf.append(name + ": " + value + "\n");
78              }
79          }
80
81          System.out.println("Handler Request: "+request);
82
83          if (request != null) {
84              buf.append("\n");
85              buf.append(request);
86          }
87
88          buf.append("</pre></p>");
89          buf.append("</body></html>\n");
90
91          response = buf.toString();
92
93          t.sendResponseHeaders(HTTP_OK, response.length());
94          System.out.println("Handler Response: "+HTTP_OK+"\n\n\n\n");
95
96          os = t.getResponseBody();
97          t.getRequestBody();
98          os.write(response.getBytes());
99
100         os.close();
101         t.close();
102     }
103
104 }
105
106 public void run(){
107   System.out.println("Running Server Thread...");
108       final InetSocketAddress addr;
109       HttpServer server = null;
110
```

```
111         addr = new InetSocketAddress(80);
112
113         try {
114       server = HttpServer.create(addr,0);
115     } catch (IOException e) {
116       e.printStackTrace();
117     }
118         server.createContext("/state", new StateHandler());
119         server.setExecutor(Executors.newCachedThreadPool());
120         server.start();
121         System.out.println("Server is listening on port " + 80);
122   }
123
124 }
```

Listing B.3: SLserver

B.4 SLClient

[56]

```
1  package client;
2
3  import java.util.*;
4  import org.apache.xmlrpc.*;
5
6  public class SLClient {
7
8   @SuppressWarnings("unchecked")
9  public void send (String url) {
10    try {
11        Hashtable theData = new Hashtable();
12        XmlRpcClient server = new XmlRpcClient("http://xmlrpc.secondlife.com/cgi-bin
              /xmlrpc.cgi");
13        theData.put("Channel", "4a2c2cfa-13ce-4a38-3072-758c38a9e4e1");
14        theData.put("IntValue", 2483); //not used
15        theData.put("StringValue", url );
16
17        Vector params = new Vector();
18        params.add(theData);
19
20        Object result = server.execute("llRemoteData", params );
21        System.out.println(result.toString());
22
23        } catch (Exception exception) {
24        System.err.println("SL_Client: " + exception);
25        exception.printStackTrace();
26        }
27     }
28 }
```

Listing B.4: SLClient

B.5 User

```
1  package filesdb;
2
3  public class User {
4
5    private String tag; //user's RFID tag hexadecimal???
6    private String name; //user name
7    private String avatar; //avatar name
8    private int uid; //Second Life id (key)
9    private String favorite;//favorite type of music
10
11   public User() {
12     this.avatar=null;
13     this.name=null;
14     this.tag=null;
15     this.uid=0;
16     this.favorite=null;
17   }
18
19   public User(String name, String tag, String avatar, int uid, String favorite) {
20     this.avatar=avatar;
21     this.name=name;
22     this.tag=tag;
23     this.uid=uid;
24     this.favorite=favorite;
25   }
26
27   public String getTag(){
28     return this.tag;
29   }
30
31   public void setTag(String tag){
32     this.tag=tag;
33   }
34
35   public String getName(){
36     return this.name;
37   }
38
39   public void setName(String name){
40     this.name=name;
41   }
42
43   public int getUID(){
44     return this.uid;
45   }
46
47   public void setUID(int uid){
48     this.uid=uid;
49   }
50
51   public String getAvatar(){
52     return this.avatar;
53   }
54
55   public void setAvatar(String avatar){
```

```
56      this.avatar=avatar;
57    }
58
59    public String getFavorite(){
60      return this.favorite;
61    }
62
63    public void setFavorite(String favorite){
64      this.favorite=favorite;
65    }
66
67  }
```

Listing B.5: User

B.6 Userlist

```
1  package filesdb;
2
3
4  import java.io.BufferedReader;
5  import java.io.BufferedWriter;
6  import java.io.FileReader;
7  import java.io.FileWriter;
8  import java.io.IOException;
9
10
11 public class Userlist {
12
13   private String FILE;//File with user's data
14   private BufferedWriter fout;
15   private BufferedReader fin;
16   private int numberOfUsers;
17   private User [] user;
18
19   public Userlist() {
20     this.fin=null;
21     this.fout=null;
22     this.FILE="Users";
23     this.numberOfUsers=0;
24     this.user= new User[5];
25     try {
26       getUserlist();
27     } catch (IOException e) {
28       e.printStackTrace();
29     }
30   }
31
32   public String [] getUserlist() throws IOException{
33
34       String [] newuser= new String[5];
35     String TMP;
36
37       boolean exists = (new java.io.File(this.FILE+".txt")).exists();
38       if (exists) {
39         System.out.println("File "+this.FILE+" found...");
40       fin = new BufferedReader(new FileReader(this.FILE+".txt"));
41       int i=0;
42       while ( (TMP=fin.readLine()) != null ) {
43         System.out.println(TMP);
44         newuser =TMP.split("--");
45         this.user[i]=new User(newuser[0],newuser[1],newuser[2],Integer.parseInt(
               newuser[3]),newuser[4]);
46         System.out.println((this.user[i]).getName()+"--"+(this.user[i]).getTag()+"
               --"+(this.user[i]).getAvatar()+"--"+(this.user[i]).getUID()+"--"+(this.
               user[i]).getFavorite()+"\n");
47         this.numberOfUsers=++i;
48
49         }
50       } else {
51         System.out.println("File "+this.FILE+" not found...");
52         fout = new BufferedWriter(new FileWriter(this.FILE+".txt"));
```

```
53              fout.write("");
54            }
55          if(fin!=null)
56            fin.close();
57          if(fout!=null)
58            fout.close();
59
60          return newuser;
61        }
62
63        public int saveUserlist() throws IOException{
64
65            fout = new BufferedWriter(new FileWriter(this.FILE+".txt"));
66            int i=0;
67          while ( i<this.numberOfUsers ) {
68            System.out.println((this.user[i]).getName()+"--"+(this.user[i]).getTag()+"--"
                  +(this.user[i]).getAvatar()+"--"+(this.user[i]).getUID()+"--"+(this.user[
                  i]).getFavorite()+"\n");
69            fout.write((this.user[i]).getName()+"--"+(this.user[i]).getTag()+"--"+(this.
                  user[i]).getAvatar()+"--"+(this.user[i]).getUID()+"--"+(this.user[i]).
                  getFavorite()+"\n");
70            }
71
72          fout.close();
73
74          return i;
75        }
76
77        public int saveUserlist(String [] data) throws IOException{
78
79            fout = new BufferedWriter(new FileWriter(this.FILE+".txt"));
80            System.out.println((data[0])+"--"+(data[1])+"--"+(data[2])+"--"+(data[3])+"--
                  "+(data[4])+"\n");
81          fout.append((data[0])+"--"+(data[1])+"--"+(data[2])+"--"+(data[3])+"--"+(data
                [4])+"\n");
82          fout.close();
83
84          return 0;
85        }
86      }
```

Listing B.6: Userlist

B.7 Music

```
package filesdb;


public class Music {

  private int index;
  private String url;
  private String songName;
  private String songType;

  public Music() {
    this.index=0;
    this.songName=null;
    this.songType=null;
    this.url=null;
  }

  public Music(String [] music) {
    this.index=Integer.parseInt(music[0]);
    this.songName=music[2];
    this.songType=music[3];
    this.url=music[1];
  }

  public int getIndex(){
    return this.index;
  }

  public String getSongName(){
    return this.songName;
  }

  public String getType(){
    return this.songType;
  }

  public String getURL(){
    return this.url;
  }

}
```

Listing B.7: Music

B.8 Playlist

```
package filesdb;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;

public class Playlist {


  private Music [] song;
  private BufferedWriter fout;
  private BufferedReader fin;
  private String FILE;
  private int numberOfMusics;


  public Playlist(){

    this.fin=null;
    this.fout=null;
    this.FILE="Playlist";
    this.numberOfMusics=0;
    this.song= new Music[20];
    try {
      getPlaylist();
    } catch (IOException e) {
      e.printStackTrace();
    }
  }

  public int setPlaylistName(String name){

    this.FILE=name;

    return 0;
  }

  public String [] getPlaylist() throws IOException{

      String [] music= new String[4];
    String TMP;

      boolean exists = (new java.io.File(this.FILE+".txt")).exists();
      if (exists) {
        System.out.println("File "+this.FILE+" found...");
      fin = new BufferedReader(new FileReader(this.FILE+".txt"));
      int i=0;
      while ( (TMP=fin.readLine()) != null ) {
        System.out.println(TMP);
        music =TMP.split("--");
        this.song[i]=new Music(music);
        this.numberOfMusics=++i;
        }
```

```
        } else {
          System.out.println("File "+this.FILE+" not found...");
          fout = new BufferedWriter(new FileWriter(this.FILE+".txt"));
            fout.write("");
        }
      if(fin!=null)
        fin.close();
      if(fout!=null)
        fout.close();

      return music;
    }

    public int savePlaylist() throws IOException{

        fout = new BufferedWriter(new FileWriter(this.FILE+".txt"));
        int i=0;
      while ( i<this.numberOfMusics ) {
        System.out.println((this.song[i]).getIndex()+"--"+(this.song[i]).getURL()+"--
            "+(this.song[i]).getSongName()+"--"+(this.song[i]).getType()+"\n");
        fout.write((this.song[i]).getIndex()+"--"+(this.song[i]).getURL()+"--"+(this.
            song[i]).getSongName()+"--"+(this.song[i++]).getType()+"\n");
        }

      fout.close();

      return i;
    }

    public Music [] getSongs(){
      return this.song;

    }

    public int getNmuberOfSongs(){
      return this.numberOfMusics;
    }
}
```

Listing B.8: Playlist

For more information on Java code visit Java Sun page[57].

References

[1] Sportv. http://www.sporttv.pt/.

[2] Arena radical. http://www.impresa.pt/NR/rdonlyres/9FB8A500-A373-4FA7-8D9B-828A5596B317/1059/PR_SICRADICAL_pt121004.pdf.

[3] Second life. http://secondlife.com/.

[4] Eyetoy. http://www.eyetoy.com/index.asp?pageID=18.

[5] Fergie's new music video. http://www.myspace.com/fergie.

[6] Steve Benford, Rob Anastasi, Martin Flintham, Adam Drozd, Andy Crabtree, Chris Greenhalgh, Nick Tandavanitj, Matt Adams, and Ju Row-Farr. Coping with uncertainty in a location-based game. *IEEE Pervasive Computing*, 2(3):34–41, July–Sept 2003.

[7] Bruce Thomas, Ben Close, John Donoghue, John Squires, Phillip De Bondi, Michael Morris, and Wayne Piekarski. Arquake: An outdoor/indoor augmented reality first person application. *IEEE*, pages 139–146, 2000.

[8] Adrian David Cheok, Kok Hwee Goh, Wei Liu Farzam Farbiz, Siew Wan Fong, Sze Lee Teo, Yu Li, and Xubo Yang. Human pacman: a mobile, wide-area entertainment system based on physical, social, and ubiquitous computing. *Pers Ubiquit Comput*, (8):71–81, 2004.

[9] Simon Prince, Adrian David Cheok, Farzam Farbiz, Todd Williamson, Nik Johnson, Mark Billinghurst, and Hirokazu Kato. 3d live: Real time captured content for mixed reality. *International Symposium on Mixed and Augmented Reality*, 10(1):115–121, November 2002.

[10] V. Lalioti and A. Woolard. Mixed reality productions of the future. *BBC Research & Development Withe Paper WHP071*, (71), September 2003.

[11] Tetsuro Ogi, Toshio Yamada, Ken Tamagawa, Makoto Kano, and Michitaka Hirose. Immersive telecommunication using stereo video avatar. *Proceedings of the Virtual Reality 2001 Conference (VR.01)*, 2001.

[12] Wolfgang Strauss, Monika Fleischmann, Mette Thomsen, and Jasminko Novak. Staging the space of mixed reality reconsidering the concept of a multi user environment. *VRML 99*, pages 93–98, 1999.

[13] Monika Fleischmann, Wolfgang Strauss, and Jasminko Novak. Murmuring fields rehearsals - building up the mixed reality stage. *Fourth International Conference on knowledge-Based Intelligent Engineering Systems & Allied Technologies*, pages 90–94, August/September 2000.

[14] Wolfgang Strauss and Monika Fleischmann. imagine space fused with data a model for mixed reality architecture. *Cast01 // understanding mixed reality*, pages 41–45, 2001.

[15] A. L. Fuhrmann, Jan Prikryl, Robert F. Tober, and Wemer Purgathofer. Interactive content for presentations in virtual reality. *VRST'01*, pages 15–17, November 2001.

[16] Oliver Storz, Adrian Friday, Nigel Davies, Joe Finney, Corina Sas, and Jennifer Sheridan. Public ubiquitous computing systems: Lessons from the e-campus display deployments. *IEEE Pervasive Computing*, 5(3):40–47.

[17] Oliver Storz, Adrian Friday, and Nigel Davies. Supporting content scheduling on situated public displays. *IEEE Pervasive Computing*, October 2006.

[18] Henrik Jernstrom. Placing ubicomp. *Ubicomp04*, pages 23–24, September 2004.

[19] Himanshu Raj, Rich Gossweiler, and Dejan Milojicic. Contentcascade incremental content exchange between public displays and personal devices. *Proceedings of the First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous'04)*, February 2004.

[20] Matthew Sharifi, Terry Payne, and Esther David. Public display advertising based on bluetooth device presence. *Mobile Interaction with the Real World - Workshop @ MobileHCI 2006*, September 2006.

[21] William T. Freeman, David B. Anderson, Paul A. Beardsley, Chris N. Dodge, Michal Roth, Craig D. Weissman, William S. Yerazunis, Hiroshi Kage, Kazuo Kyuma, Yasunari Miyake, and Ken ichi Tanaka. Computer vision for interactive computer graphics. *IEEE Computer Graphics and Applications*, pages 42–53, May/June 1998.

[22] L. Lancerica, L. Dairaine, F. de Belleville, H. Thalmensy, and C. Fraboul. Mitv - a solution for an interactive tv based on ip multicast over satellite. *2004 IEEE International Conference on Multimedia and Expo (ICME)*, pages 2159–2162, February 2004.

[23] Eric Korber and Linas Maknavicius. Bringing interactive content into the home: Dvb mhp and ip.

[24] John Fletcher and Jason White. Simutainment: A factual tv documentary as a 3d interactive experience. *IBC 2002*, September 2002.

[25] Rudinei Goularte, Edson dos Santos Moreira, and Maria da Graca C. Pimentel. Structuring interactive tv documents. *DocEng'03*, pages 20–22, November 2003.

[26] Jennifer Murdoch and George Tzanetakis. Interactive content-aware music browsing using the radio drum. *ICME06*, July 2006.

[27] George Tzanetakis. Musescape: An interactive content-aware music browser. *Proc. of the 6th Int. Conference on Digital Audio Effects (DAFx-03)*, pages 1–6, September 2003.

[28] Brad Johanson, Armando Fox, and Terry Winograd. The interactive workspaces project: Experiences with ubiquitous computing rooms. *PERVASIVE computing*, pages 67–74, 2002.

[29] Willem Fontijn and Philip Mendels. Storytoy the interactive storytelling toy. *Pervasive 2005*, May 2005.

[30] Frank Hanisch. Authoring and linking of highly interactive content within web-based courseware. February 2002.

[31] Sang Chul Ahn, Ig-Jae Kim, Hyoung-Gon Kim, Yong-Moo Kwon, and Heedong Ko. Audience interaction for virtual reality theater and its implementation. *VRST'01*, November 2001.

[32] Todd Winkler. Creating interactive dance with the very nervous system. *Proceedings of the 1997 Connecticut College Symposium on Arts and Technology*, 1997.

[33] Roger Dannenberg and Joseph Bates. A model for interactive art. *Proceedings of the Fifth Biennial Symposium for Arts and Technology*, pages 102–111, March 1995.

[34] Harry W. Agius and Marios C. Angelides. A method for developing interactive multimedia from their semantic content. *Data & Knowledge Engineering*, (34):165–187, February 2000.

[35] Alexander Repenning. Agentsheets: an interactive simulation environment with end-user programmable agents.

[36] Marco Lohse and Philipp Slusallek. Middleware support for seamless multimedia home entertainment for mobile users and heterogeneous environments.

[37] Iqbal Mohomed, Jim Chengming Cai, and Eyal de Lara. Urica: Usage-aware interactive content adaptation for mobile devices. *EuroSys'06*, pages 18–21, April 2006.

[38] Shih-Fu Chang, John R. Smith, Mandis Beigi, and Ana Benitez. Visual information retrieval from large distributed online repositories. *COMMUNICATIONS OF THE ACM*, 40(12):63–71, December 1997.

[39] Kostas E. Psannis, Marios G. Hadjinicolaou, and Anargyros Krikelis. Mpeg-2 streaming of full interactive content. *IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY*, July 2005.

[40] Giovanni Gualdia, Andrea Pratib, and Rita Cucchiara. An open source architecture for low-latency video streaming on pdas. *Ninth IEEE International Symposium on Multimedia 2007*, pages 302–309, July 2007.

[41] Jun Rekimoto. Synctap : synchronous user operation for spontaneous network connection. *Pers. Ubiquit. Comput.*, (8):126–134, April 2004.

[42] Raphael Grasset, Mark Billinghurst, Andreas Dunser, and Hartmut Seichter. The mixed reality book: A new multimedia reading experience. *CHI 2007*, April/May 2007.

[43] Mark Billinghurst, Hirokazu Kato, and Ivan Poupyrev. The magicbook-moving seamlessly between reality and virtuality. *Projects in VR*, pages 6–8, May/June 2001.

[44] Jasminko Novak, Monika Fleischmann, Wolfgang Strauss, Predrag Peranovic, and Christoph Seibert. The i2tv system: A mixed reality communication interface. 2001.

[45] Second earth. http://www.technologyreview.com/Infotech/18888/.

[46] Google earth. http://earth.google.com/.

[47] Anton Eliens, Frans Feldberg, Elly Konijn, and Egon Compter. Vu @ second life-creating a (virtual) community of learners. February 2007.

[48] Maged N. Kamel Boulos, Lee Hetherington, and Steve Wheeler. Second life: an overview of the potential of 3-d virtual worlds in medical and health education. *Health Information and Libraries Journal*, (24):233–245, 2007.

[49] Rudy P. Darken and John L. Sibert. A toolset for navigation in virtual environments. *UIST'93*, pages 157–165, November 1993.

[50] Giancarlo Fortino and Wilma Russo. The virtual video gallery: a user-centred media on-demand system. *Interactive Technology & Smart Education*, 1(1):29–40, February 2004.

[51] Stephane Natkin and Chen Yan. User model in multiplayer mixed reality entertainment applications. *ACE 06*, June 2006.

[52] Mono. http://www.mono-project.com/Main_Page.

[53] The matrix trilogy. http://whatisthematrix.warnerbros.com/.

[54] Freeview1.2. http://simteach.com/wiki/index.php?title=SL_FreeView.

[55] Second life wiki. http://wiki.secondlife.com/wiki/Main_Page.

[56] http://lslwiki.net/lslwiki/wakka.php?akka=XMLRPCImplementations.

[57] http://java.sun.com/docs/books/tutorial/.