Universidade do Minho
Escola de Engenharia

Samuel da Silva Moreira

Simulating Ubiquitous Computing Environments

Janeiro de 2013

Universidade do Minho
Escola de Engenharia
Departamento de Informática

Samuel da Silva Moreira

Simulating Ubiquitous Computing Environments

Dissertação de Mestrado
Mestrado em Engenharia Informática

Trabalho realizado sob orientação de

Professor José Creissac Campos
Professor Rui José

Janeiro de 2013

Acknowledgements

First of all, I like to thank to my supervisors. To Professor José Creissac Campos by his guidance, patience, dedication and total availability throughout the dissertation. To Professor José Rui for giving his opinion on several aspects of this work, and for helping me understand more about the Instant Places system. To José Luís for his help and availability for answering my questions and helping me understand more about the APEX framework. To all the people who work in the Instant Places, for providing information about the system. And a special thanks to Constantin Taivan for his help in carrying out some of the tests in the second case study.

A big thanks to all my colleagues and friends, and a special thanks to those with whom I lived and that helped me throughout this journey. To André Barbosa for his friendship and companionship during the course of this thesis.

And finally a thank you to my family. To my grandparents for their support during my passage through the University. To my sister for her unaware support, leading me to overcome and improve myself every day. And in particular to my parents for their support, encouragement and for doing everything they could for me, so that I can have a better life.

This work is funded by the ERDF through the Programme COMPETE and by the Portuguese Government through FCT - Foundation for Science and Technology, project ref. PTDC/EIA-EIA/116069/2009 and by FCT, under the grant with reference UMINHO/BI/59/2012.

Abstract

Ubiquitous computing (Ubicomp) technologies provide exciting new opportunities for enhancing physical spaces to support the needs and activities of people within them. The ability to develop such systems effectively will offer significant competitive advantage. Tools are required to predict problems related with use early in the design cycle.

At the Department of Informatics in the University of Minho the rApid Prototyping for user EXperience (APEX) framework is being developed. This framework allows a rapid prototyping and simulation of ubiquitous environments. The goal of APEX is to ease the creation of immersive prototypes of ubiquitous environments, so that they can be realistically explored by the users. These prototypes enable the early evaluation of how users will experience the ubiquitous environment.

This dissertation presents a state of the art in ubicomp simulation platforms. It also presents a study that defines analysis dimensions for immersive prototyping based on 3D simulation. Thus providing a framework to guide the alignment between specific evaluation goals and particular prototype properties.

The focus of this dissertation is on creating two virtual environments based on real environments, with the goal of supporting the usability testing of those environments. These tests aim to assess aspects such as people's reaction in virtual environments, assessing the ubiquitous environments created, and analyzing if these ubiquitous environments can provide a rich and desirable experience to users (based on user satisfaction when interacting with the system).

Results indicate that indeed APEX can be used to provide early feedback on the design of ubiquitous computing environments.

Keywords: Ubiquitous and Context-Aware Computing, Rapid Prototyping, 3D Virtual Environments, Evaluation, User Experience

Resumo

As tecnologias de computação ubíqua (ubicomp) oferecem novas oportunidades para enriquecer espaços físicos de modo a suportar as necessidades e actividades das pessoas dentro desses espaços. A capacidade para desenvolver sistemas eficientes irá fornecer uma vantagem competitiva significativa. Assim, são necessárias ferramentas para prever problemas relacionados com a sua utilização desde as fases iniciais do projecto de desenvolvimento.

No Departamento de Informática da Universidade do Minho está a ser desenvolvido a plataforma APEX (rApid Prototyping for user EXperience). Esta plataforma permite a prototipagem rápida de ambientes ubíquos por simulação através de ambientes virtuais 3D. O objectivo da framework APEX é facilitar a criação de protótipos de ambientes ubíquos que possam ser explorados de forma imersiva pelos utilizadores.

Esta dissertação apresenta o estado da arte em plataformas de simulação de computação ubíqua. Também é apresentado um estudo que define dimensões de análise para a prototipagem imersiva com simulações 3D. Fornecendo, assim, um enquadramento que permite guiar o alinhamento de objectivos específicos de avaliação e propriedades específicas de protótipos.

A dissertação tem como objectivo principal a construção de dois ambientes virtuais baseados em ambientes reais, tendo em vista a realização de testes de usabilidade sobre eles. Estes testes visam avaliar aspectos como a reacção das pessoas em ambientes virtuais, a qualidade dos ambientes ubíquos tal como modelados e analisar se os ambientes ubíquos criados conseguem fornecer uma experiência rica e desejável ao utilizador (baseado-nos na satisfação do utilizador ao interagir com o sistema).

Os resultados obtidos indicam que a APEX pode, de facto, ser utilizada para criar e avaliar protótipos de ambientes de computação ubíqua.

Palavras Chave: Computação Ubíqua e sensível do Contexto, Prototipagem Rápida, Ambientes Virtuais 3D, Avaliação, Experiência do Utilizador

Contents

Acronyms

3D Three-Dimensional

APEX rApid Prototyping for user EXperience

CAVE Cave Automatic Virtual Environment

CPN Coloured Petri Nets

HP Hewlett Packard

HCI Human-Computer Interaction

LSL Linden Scripting Language

PDA Personal Digital Assistant

UbiREAL Ubiquitous Application Simulator with REAListic Environments

XML Extensible Markup Language

HMD Head-Mounted Display

SUS System Usability Scale

UPnP Universal Plug and Play

List of Figures

List of Tables

Chapter 1

Introduction

Over the past few years, the interest over ubiquitous computing research has been intensifying. One of the major goals has been to shorten the gap between the technology and the humans. Marc Weiser the author of the term Ubiquitous computing (ubicomp) in 1988, claimed that:

> "The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it." [41]

considering ubicomp as an improvement over the traditional model of Human-Computer Interaction (HCI) held on desktops. The essence of this vision is the creation of environments saturated with computing resources and the ability to communicate, in a way that is transparent for the user. In these environments users become exposed to public spaces, with sensors and computing devices, allowing them to interact with such environments through personal devices, and/or by a "natural" interaction (interaction that comes naturally to human users, while interacting with the system). It is worth pointing out that more recently this vision has been contradict by some researchers, claiming that the idea of the "disappearance" of the computer is not ideal, and that what is needed is technology that captivates us (see [33] for a discussion).

Whatever the vision, ubiquitous computing technologies provide innovative solutions for enhancing physical spaces to support the needs and activities of people within them. Given their situated nature, many aspects of the interaction with ubiquitous computing systems in built environments can only be explored in the context of the target environment (the richness and texture of the actual environment is crucial to the success of the potential system). Displays, devices and sensors form an integrated whole contributing to the texture of the target environment. The ability to develop such systems effectively will offer significant competitive advantages, and tools are required to predict problems early in the design life-cycle.

1.1 Motivation

Ubiquitous computing systems pose new usability challenges that cut across all phases of design and development [36]. Additionally, the potential impact of a system in user practice, justifies that its design should be explored as early as possible [30]. When exploring and evaluating physical ubiquitous spaces, is not only necessary to explore conventional properties of usability, but is also necessary to explore properties of the environment that contribute to the users' experience. Specific situations (e.g. a crowded space or an innovative technology) need to be evaluated, and is not always possible to recreate these situations in a real-world environment. One solution is to develop physical prototypes of the ubiquitous computing system, to explore the consequences that different design decisions might have, while promoting the identification of new solutions. However the development of such prototypes might imply commitment to design decisions that would be expensive to reverse. Because of the potential cost of development and design failure, the features of such systems must be explored using early versions of the system in order to conclude whether the system could disrupt the normal operation of the target environment.

Three-Dimensional (3D) simulations can provide an alternative for an initial evaluation of the system, enabling people to experience such situations. 3D Application Servers and game engines provide a fast track to develop virtual worlds that replicate the type of environments that needs to be prototyped. The use of these resources as the basis for immersive prototyping enables the development of simulations of the ubiquitous environment. Simulated environments offer a very diverse set of properties, enabling for example different degrees of fidelity for the prototypes, from simple desktop simulations, to fully immersive experiences in a Cave Automatic Virtual Environment (CAVE) environment.

This work considers immersive prototypes for scenarios where a new ubiquitous computing system is being designed for deployment into some type of built environment. The environment will itself be a significant part of the user experience. To be successful, immersive prototyping requires a thorough alignment with the key properties of the target environment, both at the technical and social level, and a strong focus on the specific evaluation goals, such as users experience. Yet, user experience is a difficult requirement to demonstrate and assess. Obtaining information about the effects of given design can be complicated. Not only because it is costly to measure the user's feedback, but also because the relevant factors that might affect the experience must be considered. To address this, two aspects will be considered. Developing an understanding of the relevant dimensions of immersive prototyping for ubicomp systems, and developing actual prototypes to explore their value in evaluating ubcomp systems' designs.

1.2 Dissertation goals

As discussed above, the project addresses critical factors that must be considered in the construction of immersive prototypes for ubicomp systems. One is to understand the dimensions involved in prototyping this type of systems, in particular the characteristics that support the modeling of the relevant aspects of a ubiquitous environment. The other is the capacity of the virtual environment to provide sufficiently realistic simulations of the desired real environment, so that the results obtained during testing with users have relevance and can function as reliable indicators of user experience of the system in question.

With the purpose of evaluating these risk factors, our main goals are to:

- Carry out a study to relate the dimensions of prototyping with dimensions of evaluation in ubiquitous systems.
- Develop immersive prototypes of actual ubiquitous computing systems in order to:
  - Compare users' behavior in the ubiquitous system and in the virtual environments, in order to assess the capacity of prototypes to recreate the users experience of real world environments;
  - Implement new functionalities for the real ubiquitous system at prototype level in order to evaluate them with users.

This latter point will be based on the evaluation of the prototypes by users, resorting to the filling out of questionnaires, among other techniques for assessing the user experience. Initially the prototypes will developed for desktop environments, although it is also possible to use them in a CAVE [19].

To support the development of the prototypes a 3D Application Server will be used, in order to provide a fast track to developing virtual worlds and to construct simulations that can be explored realistically by users.

1.3 Dissertation structure

The dissertation document is divided into 6 Chapters. The current Chapter (Chapter 1) has presented an introduction to the motivation behind the work carried out in this dissertation. The remaining of the dissertation is structured as follows:

- **Chapter 2 – State of the art**: presents the current state of art on ubicomp prototyping approaches. First it presents relevant projects on the prototyping of ubiquitous systems focusing on isolated devices. Then it examines relevant ubicomp simulation platforms, with a particular focus on the rapid prototyping, modeling and simulations approaches. At the end of the chapter critic discussion of all the mentioned projects is made.

- **Chapter 3 – Simulation of ubiquitous computing environments**: presents an analysis of the relevant dimensions for ubicomp prototyping. A set of dimensions specific for the characterization of the prototyping of ubiquitous systems is presented, as is a set of dimensions specific for the evaluation of ubiquitous systems. Then, an analysis on how much each prototyping dimensions influences on the different evaluation dimensions is put forward. The chapter ends with a critical analysis of the research on the prototyping and evaluation dimensions.

- **Chapter 4 – Using the APEX Framework**: presents an overview of APEX framework. First it presents a detailed view of the framework's architecture. Afterwards, viewers to interact with OpenSimulator environments are presented, as are the basic steps involved in building and interacting with 3D virtual environments in OpenSimulator;

- **Chapter 5 – Case studies**: presents two case studies implemented with the APEX framework. It describes the implementation of the virtual environments, and makes an analysis on the results obtained through user testing.

- **Chapter 6 – Conclusion**: presents a summary of the work carried out in the thesis. It ends, by presenting ongoing work and directions for future work.

Chapter 2

State of the art

The prototyping of ubiquitous systems, can be said to follow two main approaches. One that is focused in the development of prototypes of specific devices in their context of use, and another that is focused in prototyping whole environments as an ecology of devices. This chapter discusses these two main approaches to ubicomp prototyping, with a particular emphasis in the second, as it is the focus of the dissertation. Section 2.1 describes approaches to develop prototypes focused on devices. Then, Section 2.2 describes several ubicomp simulation platforms, that have as goal to support the prototyping of ubicomp environments. The last section (Section 2.3) summarizes and compares the information about the ubicomp simulation platforms.

2.1 Focus on devices

When considering the progress that has been made in ubiquitous systems prototyping, it can be seen that most of the approaches are still focused on the development of prototypes based on devices. This can be easily explained with the growth in wireless devices, such as, mobile phones, tablets, situated displays, among others. Examples of prototyping frameworks for devices are the cases of UBIWISE [5, 6], Topiary [17], or UbiReal [23].

UBIWISE [5, 6] was developed at Hewlett Packard (HP) by Barton et al., and it became a reference in prototyping devices. This simulator has the goal of simulating real devices, in order to support the development and testing of context aware applications. It enables these applications to run on the simulated devices, thus avoiding their deployment in real devices. Using it, developers can create devices or interaction scenarios in significantly less time than that needed to create physical prototypes with the same features.

Ubiquitous Application Simulator with REAListic Environments (UbiREAL) [23] is a smartspace simulator, with the goal of developing and testing ubiquitous applications. These applications might have to control much information about user's preferences, state of the external environment, and others. UbiREAL generates context changes based on the user behavior and on devices communications and represent them in a smartspace (e.g. interaction between devices, changes in humidity or temperature). This approach is more interested in letting developers systematically verify the influence of context changes in an application when placed in a virtual Smartspace design, than in exploring the user's experience of the place.

The third prototyping approach, Topiary [17], is targeted at location-enhanced ubiquitous applications. More specifically applications running on mobile devices, such as smart phones. Topiary aims to support interaction designers, by providing them concrete abstractions for prototyping, such as, storyboards, scenarios and maps. The Wizard of Oz methodology [15] is used. This is a common evaluation methodology used to evaluate systems before full implementation. In this case, the methodology is used to let a developer externally control the behavior of a user's mobile phone, according to a specific planned behavior, by simulating context changes. Thus, the developer can influence the user's experience, while avoiding the use of physical sensors.

2.2 Focus on environments

Our concern is focused on virtual environments, capable of creating ubiquitous computing simulations for testing and evaluation purposes. The early construction and evaluation of physical smart environments is expensive. Additionally, the detection of construction or usability errors, while or after constructing a physical smart environment, is a major setback. Ubiquitous computing simulations give a feasible alternative to the creation of physical smart environments.

Ubiquitous computing simulation platforms can also have expensive costs associated with the evaluation and design of their applications. However, developments have been made in the ubiquitous computing simulation platforms, by making the evaluation of these applications more systematic and with lower costs [10].

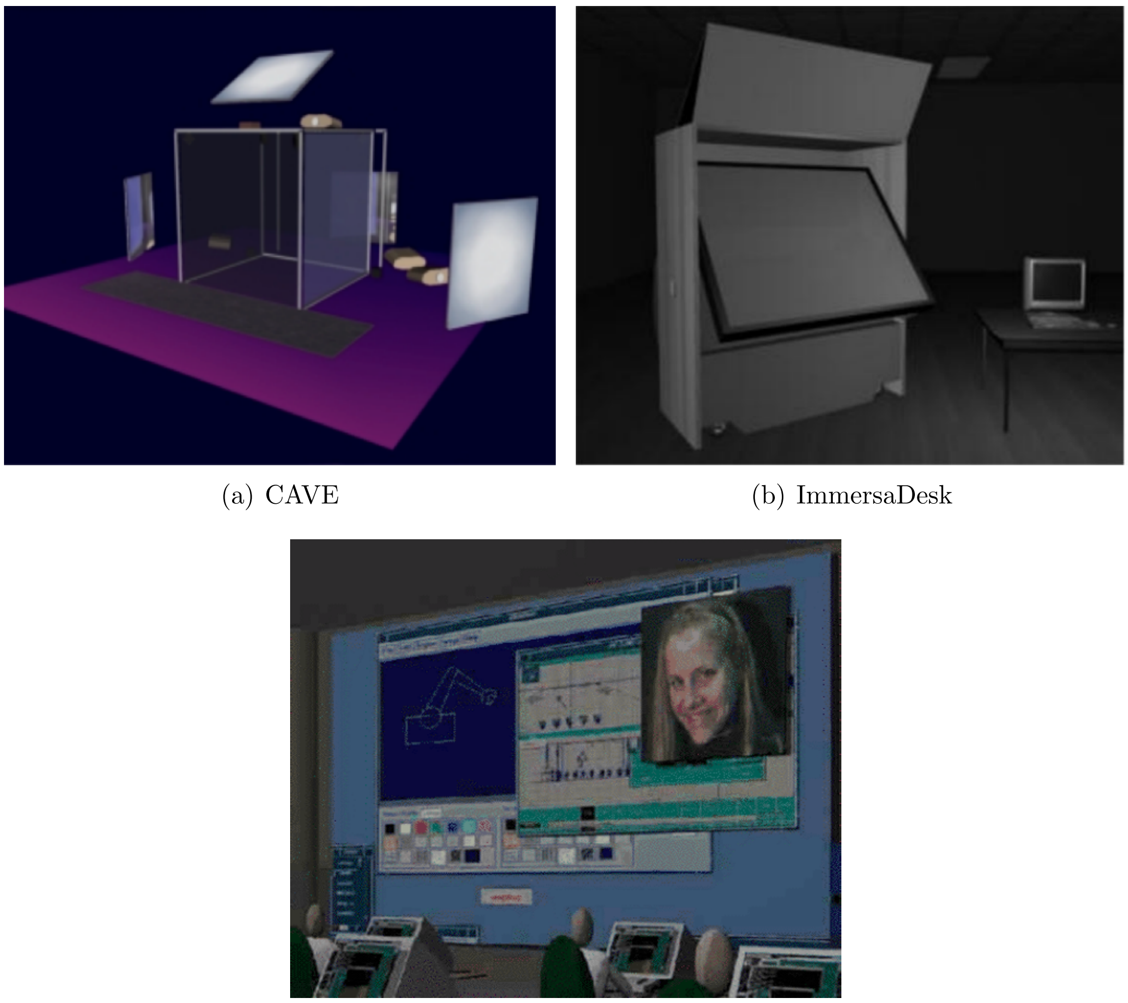
This section discusses ubiquitous computing simulation platforms, that provide a tool for developing ubicomp environments and evaluating them.

2.2.1 UbiWorld

One of the first ubiquitous computing environments simulation platforms was UbiWorld [9]. This simulator tries to combine advanced networking, supercomputing and virtual reality to explore the implications of ubiquitous computing. UbiWorld was developed in the Electronic Visualization Laboratory at University of Illinois at Chicago, the same place where the first CAVE was developed. So, in order to create a more immerse environment, and to enable the construction of 3D objects, three virtual reality devices are used to display the virtual environment. The CAVE, ImmersaDesk, and the Infinity Wall[1], can be seen in Figure 2.1.

The main goal of this project is to test UbiWorld objects and environments in a task-based way under different scenarios, where a user can explore Ubicomp-type concepts without

[1]http://www.evl.uic.edu/pape/CAVE/idesk/paper/ (Accessed: 29/1/2013)


(a) CAVE (b) ImmersaDesk

(c) Infinity Wall

Figure 2.1: Example of the three kinds of virtual reality devices used in UbiWorld.

having to build real Ubicomp hardware. To accomplish this goal some features are needed:

- **Innovative Representation Design**: take the best from the advanced industrial design and apply it to designs of ubicomp devices in UbiWorld;
- **Device/Space Awareness**: a device must show awareness of other devices and of the place where it is;
- **Transparent Networking**: network is transparent to the user. Devices are connected with each other without the user realize.

In the development of futuristic ubiquitous computing scenarios/objects some issues were identified, such as:

- **Scalability**: need to support the addition of objects into a UbiWorld environment;
- **Network Latency**: need to achieve low (user-perceived) latencies in the interactive system;
- **Object Representation**: the representation of objects must be flexible;
- **Behavior Specification**: need to identify the best way to specify the behavior of objects;
- **Object Brokering and Binding**: simplifying user interaction by making interfaces easier to use and understand;
- **Data Mapping**: the ability to map new processes (users, devices) into an existing computational framework;
- **Security**: mechanisms to control the access to information and computer resources;
- **Evaluation and Measure**: measuring, evaluating and reporting the data generated by UbiWorld.

2.2.2 3DSim

3DSim [21] is a tool for the rapid prototyping of Ambient Intelligence application. The motivation is to test ubicomp applications in a virtual environment, before deploying them in the real world. Additionally, to test the same ubicomp application in several different domains (meeting room, home). Figure 2.2, shows a example of a virtual environment using 3DSim.

The 3DSim tool allows developers to concentrate on:

- **Human-ambient-interactions solutions**: It provides realistic impressions of the visualized domain, through the use of interactive 3D visualizations of a virtual domain;

- **Strategy Components and Re-active Agent Systems**: Using it developers can validate and test the dynamic behavior of several components and agents;
- **Situation-Recognition**: It has an environment monitoring system (EMS), responsible for analyze sensor data and for detect higher-level environment state transitions;
- **Dialogue Management**: 3DSim enables use of 3D gestures to point to devices and room objects.
- **Actuator and Sensor Integration Interface**: 3DSim core system use TCP/IP-based eventing interfaces, that integrates a 3D scene changed by Universal Plug and Play (UPnP) actuators and sensor components.

The use of a photo-realistic and interactive 3D virtual environment, enables users and developers to get a realistic view of the environment, allowing a first judgment on the usability and system adaptability to a place. Integrating a ubiquitous system in a new place allows testing the suitability of the system to different domains (e.g. house, meeting room, and others), before actual deployment in a physical environment. Additionally, it allows the observation of the behavior of the application under different interactions by the users. Thus, this approach is more focused on the application functionality, than on the users' experience.

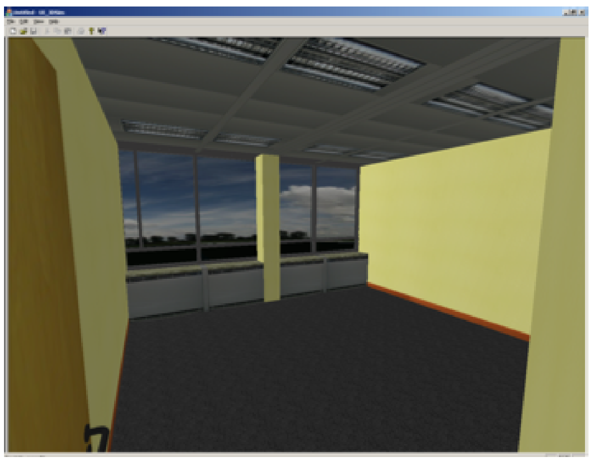
3DSim has the flexibility to manipulate system settings and conditions, and even change the state of devices, allowing the possibility to observe the system's reaction to these changes. The possibility to manipulate objects (state, position, etc), change the position and intensity of lights, and the temperature in a room, are examples of benefits of using this tool, and a way to save costs and speed up development cycles.

For situation recognition (changes in sensors, etc.) and to simulate sensor data, 3DSim uses an Environment Monitoring Subsystem (EMS). It analyzes the sensor data, and if necessary triggers a (visible) reaction in the virtual environment. Automatically changing the state of the shutters (open/close) according to the light settings, is an example of this capability.

3DSim also allows persons to use 3D gestures to interact with devices and objects in the virtual environment. Providing these accurate data about user's gestures, supports the development of dialog management systems.

The architecture of 3DSim is essentially a CVE-server (Collaborative Virtual Environment) and 3DSim-clients. The CVE-server manages the 3D environments and stores the state of every object in the environment. Meanwhile the 3DSim-client, provide a interface for the user to explore the 3D environment.

3DSim provides a set of devices and actuators with full UPnP support, which can be dynamically integrated in the virtual environment, by dragging the corresponding Extensible Markup Language (XML) description-file in the 3D scene. The CVE is based on a system that manages object life cycles and provides event distribution in the virtual environment. When the CVE-Server receives the messages, it updates each affected entity in the virtual environment, and then reproduces the 3D view at the client side. For rendering purposes, 3DSim uses the RenderWare Plataform [4] for game development.


(a) 3D rapid prototyping environment.

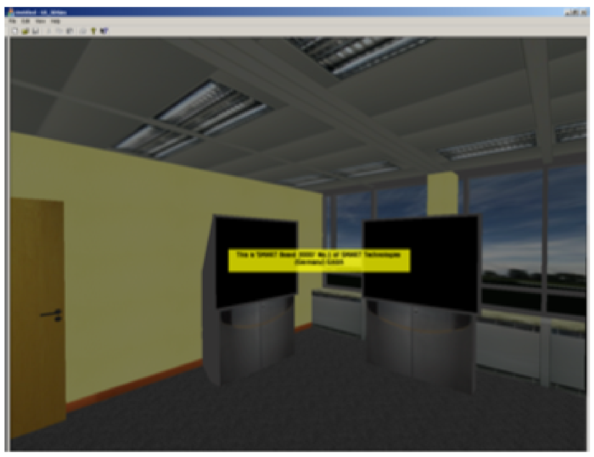

(b) Environment with three displays.

Figure 2.2: Examples of 3DSim environments.

Another important operating mode in 3DSim is the context visualization mode. It allows the analysis of events received from real sensors and their representation and animation in the virtual environment. This is used to test the precision in context aware systems, using sensors to recognize any sort of activity and trying to reproduce into the 3D environment the state of that activity. Examples of this are, when EMS recognizes that a real person is sitting in a chair with pressure sensors, or when it detects a person cleaning/writing in a whiteboard, 3DSim animates an avatar to reproduce this actions. This is also true of environmental and devices changes (turning lights off/on, displaying device states, or opening/closing doors). These changes are also rendered in the 3D environment. With this, developers can test the fidelity and responsiveness of the context aware system, or reconfigure the environment to react suitably to the events.

Overall, 3DSim have features which makes it a flexible, extensible and a different tool from others approaches. Enables rendering scenes to give photo-realistic impression to the 3D environment and supports the development of human-ambient-interaction systems or adaptive user interfaces. Another difference is that it allows the interaction of virtual sensors or real sensors connected through UPnP. Using this standardized interface means that it can be easily adapted to real world environments. To conclude, 3DSim is aimed mainly to help developers, developing realistic and controlled environments, and user interfaces. The possibility of using either real or simulated sensors, gives a richer user experience to users and developers.

2.2.3 TATUS

A high-level goal of the TATUS project [24, 25] was to provide a suitable and flexible 3D virtual environment, that allows developers to test ubiquitous computing applications under development. Developers/designers can use it to test and evaluate applications in a 3D virtual environment without having cost and logistic issues. It can be used to understand if there exist any unwanted causal relation, when a user makes a decision

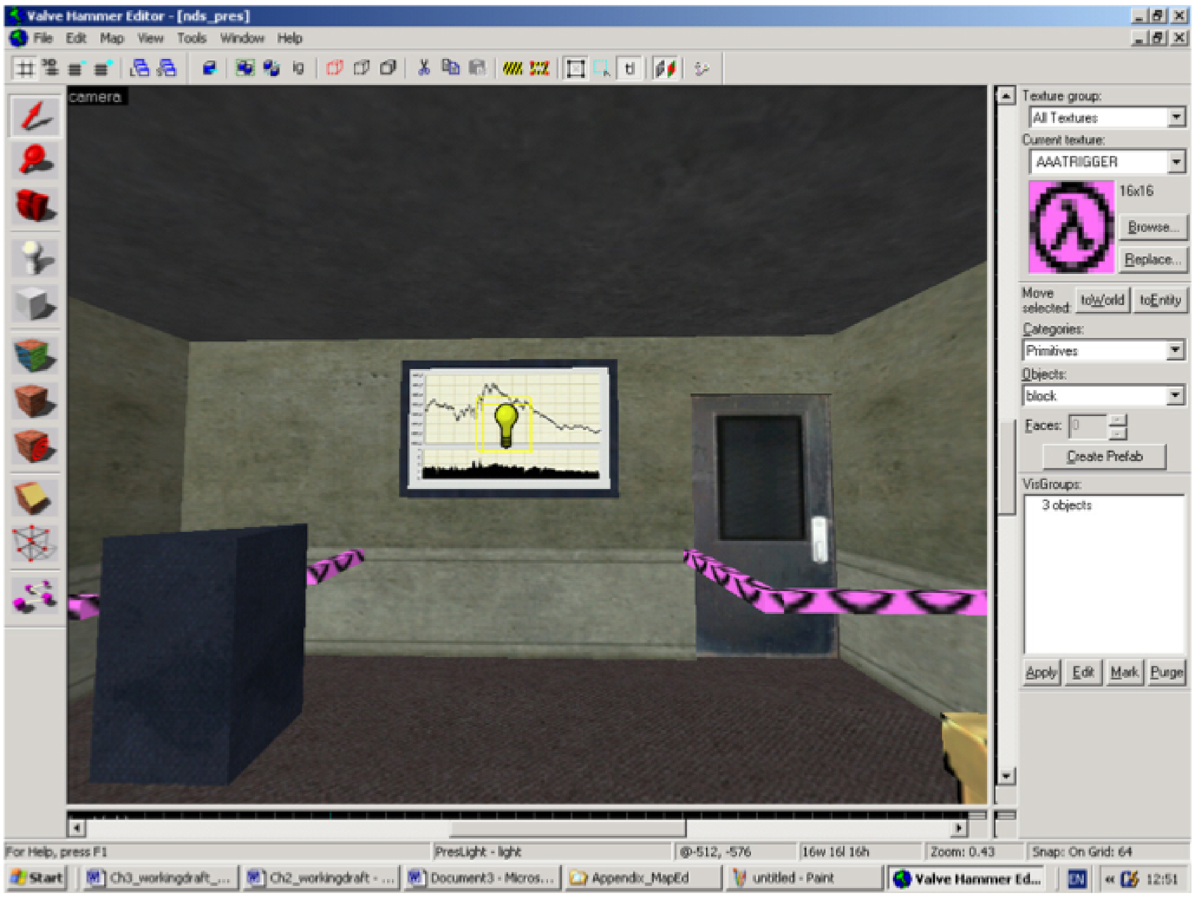


Figure 2.3: Map editor's rendering of an environment

that triggers some unexpected behavior in the virtual environment. A software-under-test (SUT) connected to the simulator is used. The SUT can make decisions and adapt its behavior according to environmental changes, or in reaction to user behavior (actions, position, etc.).

The virtual environment representation is achieve through a 3D game engine developed by Valve Software, and used in many of its games, e.g., Half-Life 2. The main reasons behind the choice of this 3D game engine are, the fact that its SDK is available for free, and that it provides a high customization of the virtual environment and physics simulations, enabling the creation of realistic environments. Valve Software also provides tools used for building the 3D virtual environments, as can be seen in Figure 2.3.

The main features of the simulator are:

- **3D Graphical Interface**: Use of a 3D interactive graphical user interface, to allow multiple and simultaneous users to explore and test the virtual environment, as shows Figure 2.4;

- **Separation of simulator and SUT**: The SUT and the simulator are physically separated. This allow simultaneous connections to multiple SUTs.

- **Realism**: It enables the development of realistic virtual environments. Simulating actual sensor data, instead of exploit the simulator data.

- **Flexibility**: The simulator supports the test of a range of software. It is generic, so it can provide a specific state or interface a particular piece of software.

- **Usability**: The quick, simple and flexible implementation of an experiment is achieved through the use of existing map editors and an additional message definition tool.

- **Extensibility**: The underlying SDK can be adapted to extend several features offered by the simulator framework.

- **SUT API**: Selection and extraction of state information is provided to developers. The SUT is also capable of making decisions and actions on the virtual environment. The researchers can set up a test environment, most fitting for the SUT experiments.

In conclusion, TATUS is a flexible ubiquitous computing simulator for use as a tesbed for SUT. TATUS is focused on controlling ubiquitous computing environments through the use of sensors and actuators. By using sensors and actuators, it tries to predict user intentions in the virtual environment. It is also focused on the behavior of adaptive ubiquitous computing applications that are centered on the user. Regarding users, TATUS tries to make them feel immersed in the 3D virtual environment through the whole experiment. For researchers, TATUS guides them through the stages of developing a scenario, framing all test situations for running the test successfully.

Figure 2.4: Simulated meeting room scenarios showing other characters

2.2.4 VARU

The VARU framework [11] aims to integrate, in a single platform, a rapid prototyping framework for augmented reality, virtual reality, and ubiquitous computing spaces (environments), as can be seen in Figure 2.5. It uses OpenSceneGraph[2] as the game engine to render each of these spaces. It is possible to interact and collaborate with users that exist in different spaces, and it is also possible to change from an interaction space to another, without much effort.

Different users can interact with the same object, each using a different representation. I.e. a user in a virtual reality space can interact with the virtual representation of the object, while a user in an augment reality space can also interact with the physical representation of the object. This raises the problem of how to synchronize the same object with the different representations present in the different interaction spaces that share the object. To solve this, three levels of abstraction are used to describe an object: (Object Class, Individual and Extension).

[2]http://openscenegraph.org (Accessed: 29/1/2013)

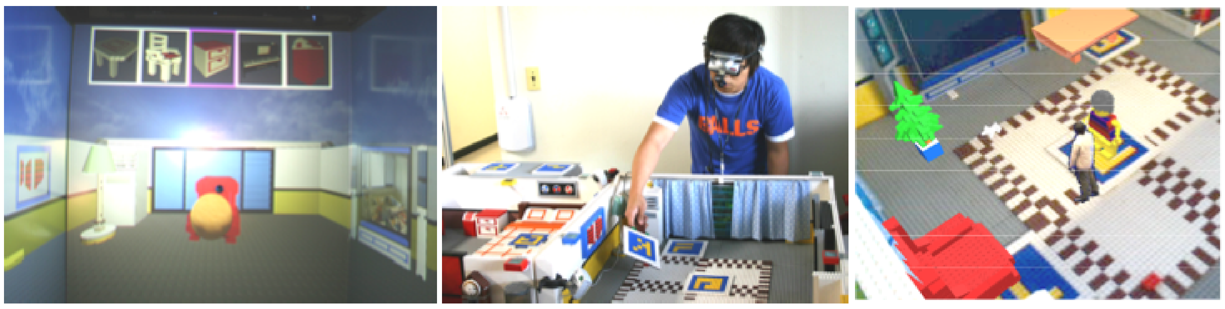


Figure 2.5: Different perspectives on how to interact in VARU framework.

- **Object Class**: This represents a abstract group or a specific collection of objects. For example, Table is the class of all tables.
- **Individual**: This represents an instance of an Object Class. For example a Black_Table is a instance of the Class Table
- **Extension**: This represents an Individual in a specific interaction space. For example the Black_Table@Client1 is an Extension of the Individual Black_Table.

So when a user joins a AR space or a VR space, new extensions to the existing Individual objects are created for that interaction space (e.g. the extensions Black_Table@AR or Black_Table@VR can be created for the individual object Black_Table). This is all stored in the Object Database, which is one of the three components of the VARU Server, that supports the synchronization of objects throughout the different interaction spaces (e.g., when a user in a interaction space changes the values of a object, their attributes' values must be updated on the Individual objects). Later, the extensions of the individual objects in the different spaces are also updated. Thus, users become aware of each other's actions. The actual responsibility of keeping the objects synchronized, belongs to another component of the VARU Server, the Object Server. The last component is the Simulation server, which is responsible for simulating the events in the virtual environment.

The VARU Client must have a kernel to make the communication between the VARU Client and VARU Server possible. The VARU Client have a Space Manager (AR/VR/UC

Manager), responsible for managing its interaction space. The other components are the Device Manager, Display Manager and Streaming Manager. The Device Manager is responsible for managing the input/output of devices. The Display Manager is responsible for configuring the display system of VR and AR spaces.The Streaming Manager is responsible for streaming images through the network

So, the VARU framework differs from other approaches due to the fact of supporting a mixed space collaboration. Application developers can design applications involving a virtual, physical or mixed spaces, according to their available resources. Allowing users the possibility to interact with virtual and physical objects whatever is their location. All of this, contributes for developers to have a easier and efficient way to develop tangible space applications.

2.2.5 APEX

The APEX (rApid Prototyping for User eXperience) framework [37], which is being developed at the University of Minho, allows for the rapid prototyping and simulation of ubiquitous environments. The approaches of the previous simulation platforms focus mainly on testing/analyzing the software/devices that are being developed, and how they react to context changes. In general, they do not have support for testing environments with many concurrent users (an important aspect ubiquitous environments). The APEX framework seeks to answer these two problems.

In order to discover possible issues in a ubiquitous environment's design, we must conduct evaluations from the early stages of design on how the users will interact with the environment, and about their reactions to that experience. The main goal of the APEX project is to facilitate the creation of prototypes of actual ubiquitous environments, allowing the users to interact with the virtual environment in such a way that the experience feels natural to them. The APEX framework generates prototypes and their simulation enabling users to navigate through them to evaluate usability issues. At the same time it will help the developer to understand how user might experience the system.

To construct virtual worlds that might be explored realistically by users, a system which enables their creation is necessary. In particular, 3D application servers offer tools for construction and navigation through virtual worlds, which allow the creation of objects, such as, characters, houses and terrain. They also let different users (avatars) connect to the 3D application server and interact with each other. Furthermore, 3D application servers provide features to add behavior to objects through scripts. Additionally, with the use of model based techniques, it becomes possible to specify the behavior of the whole system at a higher level of abstraction, and subsequently make a exhaustive analysis of that systems' behavior.

The APEX framework uses OpenSimulator[3] as its 3D application server. The main reason for this is because it is open source, which means that the server is programmable, allowing greater configurability and extensibility. For modeling ubiquitous systems, the APEX framework uses Coloured Petri Nets (CPN) [14], an extension of the Petri nets [28] modeling language. Additionally, the support for formal analysis through the use of CPN Tools[4] is another main advantage. OpenSimulator and CPN Tools working together allow for the rapid prototyping of the envisaged ubiquitous environment. They support both users to experience the virtual environment and behavioral analysis.

APEX features are supported by four components which compose the APEX architecture as shown in Figure 2.6:

- **Virtual Environment Component**: responsible for managing the layout of the virtual environment through a 3D application server (Opensimulator);
- **Behavior Component**: responsible for the description, the analysis and validation of the virtual environment's behavior, using CPN to model sensors and dynamic objects, and also responsible for the animation of the behavior;

[3]www.opensimulator.org
[4]cpntools.org

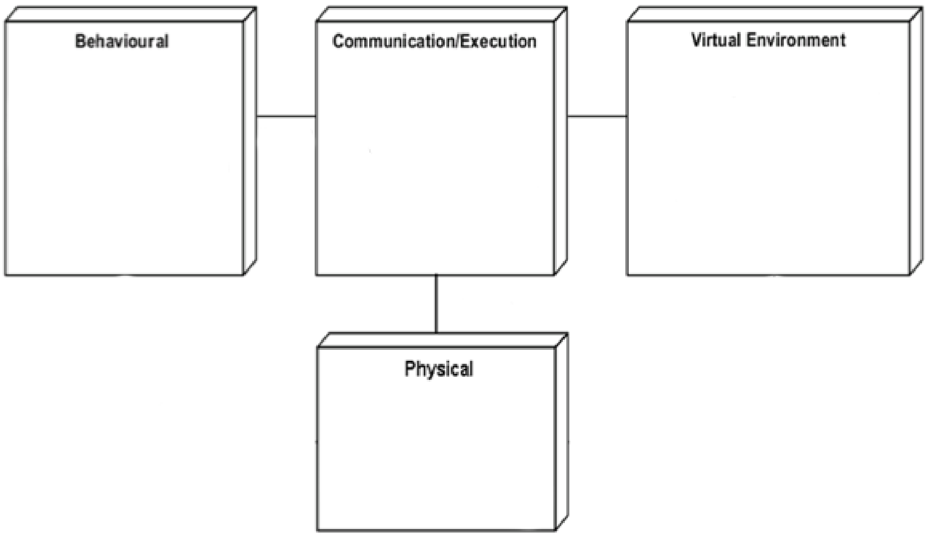


Figure 2.6: Simple example of APEX framework architecture.

- **Physical Component** responsible for the connection of external physical devices (e.g. smart phones, sensors) to the virtual world via Bluetooth;
- **Communication/Execution Component**: responsible for the data exchanges between all components, while the simulation is running, and for the execution of the framework.

To explain how the APEX framework works, and to demonstrate the role of each component, an example with a smart library will be used, as shown in Figure 2.7. This library features sensors placed close to a gate (A), books (B) are identified by RFID tags and are stored in bookshelves (C). Screens are used to provide information to users. The main goal is to help users locate books through the use of sensors to recognize the user's position in real-time. In particular we will demonstrate how the gate of the library works, taking into account the position of the user. The virtual environment component sends information from the simulation to the behavior component and manages which objects must be seen and their characteristics (e.g. position in space). The behavior component is responsible for receiving the information (sensors value) coming from Opensimulator, making decisions

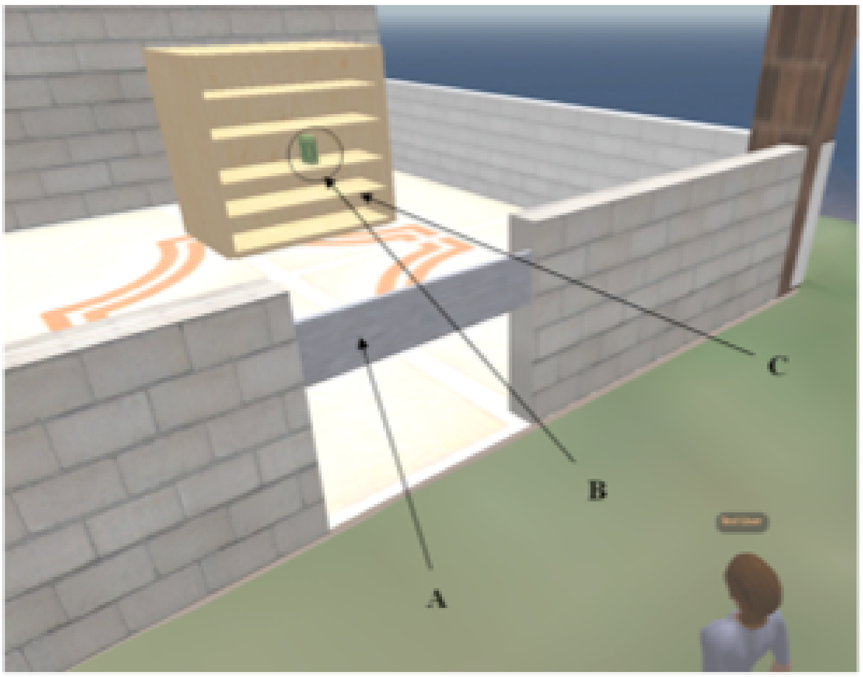


Figure 2.7: Example of a library simulation

(open/close the gate), reflecting the CPN models, and sending relevant data to the simulation. In other words it has the function of managing the behavior of objects in the virtual world, which in our case can be considered as the opening or closing the gate of the library.

In order to be able to run the simulation, and have the virtual environment adjusted to the users' navigation and interactions, a communication/execution component, able to link the previous components, is needed. Finally, the physical component is used in this case to connect devices such as a smartphone or a Personal Digital Assistant (PDA), in order to provide directions to users. In the example of Figure 2.7 the virtual component is responsible for sending information to physical devices and to the modeling component, and for showing the virtual environment. Once the user moves his avatar towards the library gate, the behavior component (CPN model) will receive the sensors' information about the presence of the avatar in the space, and check if its position is near the gate. If it is not near the gate nothing happens, otherwise the CPN model makes the decision of opening the gate, and then the action is reflected in OpenSimulator. After the avatar enters the library and moves away from the gate, the CPN model checks if the avatar is already far from the gate, if it is the decision of closing the gate is made and the action is reflected to OpenSimulator. All of these information exchange between OpenSimulator and CPN model are controlled by the communication/execution component.

2.3 Conclusion

We will use the APEX framework to develop virtual environments representing a specific physical space augmented with ubiquitous technologies, and to evaluate user related aspects of the proposed design (experience, usability). Table 2.1 shows comparisons between the ubicomp simulation platforms identified in the previous section. In relation to UbiWorld and 3DSim the main advantage of APEX is that it uses models for creating the envisaged systems, unlike the UbiWorld and 3DSim approaches that use programming languages to do it. The benefit of this is the possibility of combining a model approach with analytical approaches, providing a leverage on properties of ubiquitous environments that are relevant to use. The main advantage of APEX framework to the work of O'Neill [25, 24], is that APEX provides more flexibility and a multi-layered prototyping approach supporting different levels of analysis. Enabling developers to verify system properties and allowing them to move between, and evaluate specific features of, different layers. Others advantages that APEX has in relation to the mentioned simulation platforms and to VARU, is the use of a 3D application server instead of game engines and the possibility to have exhaustive analysis support. The use of a 3D application server provides a few benefits as, the possibility of creating and customization a virtual environments in real time, the possibly of loading modules, and support for multiple users to access the virtual environment at the same time. Using an exhaustive/formal analysis, gives the possibility to analyze every system behavior and the possibility to prove specific properties [37].

In conclusion, APEX gives us all these advantages in relation to the ubiquitous computing simulation platforms previously mentioned. Hence, as stated above, APEX will be used to build the virtual environments that we propose to make.

	UbiWorld	3DSim	TATUS	VARU	APEX
Ubicomp environments prototyping	yes	yes	yes	yes	yes
Provide user experience	yes	yes	yes	yes	yes
3D application server or game engine	game engine	game engine	game engine	game engine	3D application server
Multi-layered prototyping approach	no	no	no	yes	yes
Programming or modeling approach	program-ming	program-ming	modeling	N/A	modeling
Exhaustive analysis sup-port	no	no	no	no	yes

Table 2.1: Distribution of codes by frameworks

Chapter 3

Simulation of ubiquitous computing environments

To better understand how virtual worlds might support the prototyping of ubicomp environments, we need to establish how to align them with the key properties of the target environment, as well as the specific evaluation goals that they should support. Particularly, we are interested in spaces enhanced with sensors, public displays and personal devices, and in understanding how prototypes support evaluation of such systems. This chapter defines analysis dimensions for immersive prototyping based on 3D simulation, and provides a framework to guide the alignment between specific evaluation goals and particular prototype properties. This should provide a relevant contribution towards understanding the potential added-value of 3D simulation as a tool in the development process of ubiquitous computing environments. The key issue that we want to address is "what are the relevant dimensions that prototypes should exhibit to better support evaluation of the envisaged design ?".

A similar process to the one described herein was carried out by Ostkamp et al. [27], but in that case the authors were only interested in studies about public displays. They introduce the AR-Multipleye, a system that visually highlights items on a personal device that is pointed towards a public display, and then evaluate existing highlight methods for public displays according to a set of classification criteria. As said above, we follow a similar approach but focused on the immersive prototyping of ubiquitous systems.

In order to establish relevant analysis dimensions for ubicomp immersive prototyping, the chapter performs a review of the literature on the topic. The collected papers address several topics about ubiquitous computing, with a specific focus on virtual environments. Several groups of papers are identified, with each group addressing a specific ubiquitous computing topic.

Most of the papers are related to the rapid development and evaluation of ubiquitous systems in the early stages of the development life cycle. Examples include 3DSim [21], TATUS [24, 25], the work of O'Neill et al. [26], UBIWISE [5, 6], the work of Reynolds [31] or APEX [37, 34]. Others papers, as UbiWorld [9] and the work of Pushpendra et al. [38], are focused in creating immersive environments for users, and testing their applications, using CAVEs and other immersive technologies. VARU [11] and CityCompiler [20], UbiREAL [23], and the work of Brandherm et al.[29], focus their study in hybrid prototyping approaches, integrating services (e.g. Internet services) and devices in their ubiquitous systems. A few papers are more concerned with the analysis of user behavior when confronted with different situations (this is the case of Siafu [18] and the work of Maly et al. [13]), while Topiary [17] and the work of Li et al. [16] are more concerned with the context awareness behavior of ubiquitous applications. In particular Topiary enables developers to use the Wizard of Oz methodology to control the experience of using a mobile phone.

3.1 Methodology

The papers were analyzed in search of codes for two groups of characteristics of ubiquitous computing that we initially defined as:

1. Properties of the simulation;
2. Evaluation requirements and objectives.

Open Coding [39] was used to analyze the contents of the papers. Each paper was read in order to identify phrases or paragraphs containing references to the two groups of characteristics of ubiquitous computing aforementioned. A code was assigned to each piece of text identified. At this stage, the goal was to generate as many codes as possible without much consideration of how they related with each other. The MAXQDA10[1] tool was used to aid the open coding process. A total of 33 different codes were identified. Of these, 20 corresponded to the first group, while the other 13 related to the second group. A total of 220 instances of codes were identified.

An affinity diagram[2] was created to synthesize the gathered data (in our case codes). The goal here was to find the key dimensions, based on the natural relationships between codes. In a brainstorming session we grouped similar properties into logical groups. As we analyzed more codes, we discussed whether to place each of them in one of the existing groups, the possibility of creating more groups or of creating subgroups.

As the results of this process, we identified two distinct groups of characteristics of ubiquitous computing. The first group characterizes the relevant features of immersive **prototyping** ubiquitous systems. The second group characterizes the different perspectives on ubiquitous systems **evaluation**, and the methods used to gather feedback about user experience. The list of dimensions (and sub-dimensions) in each group is presented below.

Prototyping

- Fidelity of immersion
- Embodied interaction support
- 3D modeling and simulation
- Controlled environment manipulation

[1]http://www.maxqda.com/products/maxqda10/ (Accessed: 29/1/2013)
[2]http://infodesign.com.au/usabilityresources/affinitydiagramming/ (Accessed: 29/1/2013)

- Context driven behavior
- Multi-user support
- Hybrid prototyping

Evaluation

- Controlled experiments
- System-centric Evaluation
  - Developer-centric evaluation
  - Environments evaluation
- Data collection
- User-centric Evaluation
  - Evaluating user experience
  - Evaluating usability

3.2 Dimensions features

This section presents the groups identified in the previous section and their dimensions. It provides a description of each of the dimensions and, where relevant, presents specific cases that were identified in the analyzed papers.

3.2.1 Prototyping

The Prototyping group of dimensions captures the relevant features of a system for the immersive prototyping of ubiquitous computing systems.

Fidelity of immersion

Fidelity of immersion, in this context, can be described as the possibility to represent the real world in a virtual environment. Specifically, the better the virtual environment represents the real environment and the better the user feel submerged in terms of appearance, sound and interaction, the better he feels immersed in the virtual environment. Creating this type of environments is beneficial to the user, because it creates a closer connection between him or she and the authenticity of the environment. However, it can be difficult and time consuming to recreate these virtual environments.

A number of techniques exist with the purpose of immersing users when presenting them the virtual environment. These techniques go from the use of head-mounted displays or augmented reality to mix virtual information with real environments, to the use of CAVEs [7] (see, for example [38, 9]), or other CAVE-derived techniques as presented in [9], the ImmersaDesk and the Infinity Wall [8].

An example of immersion is the case of immersive video inside a CAVE. This approach eases the evaluation and prototyping of mobile applications before its actual deployment, providing a high fidelity recreation of a user's experience [38].

3D modeling and simulation

3D modeling and simulation is a means to build virtual environments/devices and also to simulate them. This is typically achieved through the use of game engines or 3D application servers. This is also a key factor to make the virtual environment realistic, by providing e.g. better rendering, better physics and a better response to collisions. It should be noted however, that creating a realistic simulation extends beyond the physical and graphical qualities of the simulation. For example, [5] points out that creating a realistic simulated wireless device, implies being realistic in terms of connection latency, bandwidth, screen size, and battery life.

According to [26] the use of game engine allows for a greater flexibility in the type of sensors that are used. The most used game engines in the construction of virtual environments are Half-Life, Unreal, and Quake. In [37] the Opensimulator 3D application server (an open source alternative to SecondLife) is used. According to the author one advantage of using a 3D application server is to enable the remote and simultaneous connection of many users over the internet.

Embodied interaction support

This type of interaction refers to the ability of the ubiquitous systems to enable the reproduction of interactions that we use every day in the real world, in a natural and intuitive manner. Embodied interaction can be achieved through the use of a few interactions technologies, such as motion tracking and gesture or speech recognition, to improve the usual ways in which user actions are executed (mouse, keyboard). Users may, for example, interact with the virtual environment through the use of 3D gestures to point to devices and room objects [21]. In particular, gestures can be used to control interactions within the virtual environment, allowing a more interactive and immersive experience. In [20] a scenario is built where a camera captures the size and location of human shadows and, based on that, triggers appropriate events (e.g. displaying a video).

Controlled environment manipulation

Ubiquitous systems' simulation can be molded to best serve the objectives of the designers and developers. We can define the behavior of the system and its objects, by programming them, by the use of models, or we can manually control/influence this behavior, e.g. trough Wizard of Oz techniques.

The most common method, for expressing behavior is programming it through the use of scripts. Usually, scripts store all system settings, enabling that several users might have the same scenario, with the same settings, in their individual sessions. With this, developers have a easy way for testing and comparing the metrics that they want to evaluate [5, 37].

Another approach to attach behavior and functionalities to the system and its objects, is through the use of models. We can model the behavior, whether of the system or of the objects, so that they can simulate events in response to context changes or user actions. In [37] and [26], models are used in combination with a 3D simulation to prototype virtual environments. More specifically, in [37], CPN [14] models are used to describe the behavior of the objects in the virtual environment.

Wizard of OZ can also be used to give behavior to the system. Li et al. [16] use this method to simulate the use of sensors in testing context-aware applications, avoiding the costs associated with real sensors' deployment. The need to use people to realize experience tests, and the fact that these tests are never realized in the exact same circumstances, are problems associated with the technique [38].

Context driven behavior

Context driven behavior happens when the system/prototype is able to capture the state of the environment and its relevant data, adjusting its behavior to that data. These systems need to learn the skill of constantly adapting to context changes (e.g., a door opens, when a user gets close to it). This feature is present in many systems [17, 25, 9, 26, 5].

Approaches to gather context data are e.g., sensors, devices (e.g. GPS readings [16]), systems with information about networks, or specialized tools to extract information from the virtual environment [21, 13]. Sensors, in particular, are very common in ubiquitous systems. According to [31], sensors can be classified as active or passive, i.e., they can detect values internally or from the virtual environment, respectively. Sensors can act as listeners for the system, enabling it to react to the environment [21] and storing relevant sensor information for later use [38].

Multi-user support

Enabling multiple users to explore the ubiquitous system allows for faster testing and assessment of the behavior of the system. Two approaches to consider scenarios with multiple concurrent users in a single experiment are: supporting the connection of multiple real users, or supporting the use of bots in the system.

Supporting multiple real users allows mainly to evaluate their behavior and their interactions in the system. At a second level, it allows evaluating the behavior of the system. In [37, 26, 25], this is an important dimension to integrate in the development of the ubiquitous system.

Supporting the use of bots (i.e. AI expert software systems), allows essentially to evaluate the behavior of the system and its functionalities. Bots can also have the ability to emulate the decision-making ability of a human being. Bots can be used to support the configuration of multiple user environments with a limited numbers of real users, or to systematically explore an environment (e.g. to automatically identify unwanted behaviors) [26, 25].

Hybrid prototyping

Hybrid prototyping can be defined as an approach that takes advantage of the use of a combination of simulated and real components when building the prototype. This generates a mixed reality where augmented reality can be used to compare the simulation and the experiment. In [20], a mixed of physical miniature prototyping and virtual prototyping is used. This type of prototyping can make software more flexible, robust and allow an easy integration with the interactive systems in the environment.

On this topic, we covered two types of hybrid prototyping, one that is specific for devices and other specific for services. We start by presenting examples of the hybrid prototyping of devices. Virtual devices are an approach used to accomplish this goal. These devices can be simulations or emulations (recreation of the original look and behavior) of actual devices, e.g., smartphones, PDAs, sensors. In [5], images of the device's physical interface are used to create the virtual device. The use of real/simulated sensors can enable testing specific systems, and their integration in the ubiquitous environment, without actual physical deployment. The embedding of sensors in virtual devices, e.g., to send a signal of the user estimated position or show their location in a device via a 2D map, is addressed in [29]. A emulation framework allows testing the applications, and allows that simulated hardware devices can interact with the emulated software [31].

Hybrid prototyping of services can provide a higher realism, accuracy and precision in ubiquitous systems, since it can use real services, e.g. internet access. Adding real/simulated services to ubiquitous systems and allowing users to exploit them, enhances the system's functionalities and can provide increased user satisfaction when interacting with it. The most common cases are the integration of internet services, or the use of Bluetooth service to integrate real devices, or the use of similar protocols with suitable bandwidths for supporting the communication between them and simulated components [37, 5]. Many systems tend to create their own communication components, using protocols such as TCP-IP or UPnP [21], or resorting to proxies [25], while other systems integrate existing network simulators into their framework [31]. The advantages of having this integration are, e.g., not having additional costs, and giving to users and developers a more enhanced experience.

3.2.2 Evaluation

Evaluation is a key motivation in the immersive prototyping of a system. In the current case, two types of evaluation interests could be identified. Evaluation focused on the system and its developers, and evaluation focused on the potential users of the system. First, several examples of how to conduct controlled experiments and also different ways to collect evaluation data (user feedback, user experience) are presented. Then, a description of the evaluation dimensions identified in Section 3.1 is presented.

Controlled experiments

Before a ubiquitous system is deployed in the real world, it should be subjected to exhaustive interaction tests by users, under varied environment settings and context changes. Controlled experiments can be performed to achieve this. An approach is to replicate the exact same experiment with different users. As discussed above, a way to do this is through the use of scripts. All experiments should have the same system configurations, e.g., the events generated by sensors or the way the system adapts to context changes must be the same for any user that interacts with ubiquitous system.

Another approach, is to change one or several ambient settings in each experiment. This leads to an understanding of the possible reactions that can happen when the system is placed in a real environment. These manipulations can go from re-positioning objects and avatars, to the manipulation of actuators and devices such as, lights, temperatures or displays, and changes of virtual device states [21, 37, 23]. The more common examples that were found in the literature were to manipulate lights and temperatures, in order to see how these changes affect the system. In particularly in [37], it is mentioned that all of these manipulations, enable creating a more realistic visualization of the proposed real system.

It is also necessary to carry out experiments to observe the behavior of the system when deployed in different scenarios, or to observe how the system reacts through time, when events that happen in the real world are simulated. Particularly regarding time, changes of context can directly or indirectly influence the action, and even change the behavior of a user or a object that is subject to assessment. Increasingly, ubiquitous systems are being developed to function and adapt to different scenarios [21, 9, 37]. In [21], the authors evaluate the suitability of a device to new environments and their adaptability to different interactions, more specific, the Philips iPronto device.

Data collection

Data Collection is a increasing concern in the evaluation of ubiquitous systems, so an important question comes up, "which methods to use to evaluate the users and which methods use to gather the feedback of their experience?". Developers can gather user feedback, either by allowing the user to freely explore virtual environments, or by making him or she follow or perform a list of tasks and storyboards, e.g., making the user complete a series of tasks in a virtual device, such as, order a meal, make a phone call, prepare a presentation [9].

There are several methods to perform evaluation tests and to collect data from these evaluation tests [3, 2]. Next, the methods that appear more often in the studied papers, and the ones that we think are more relevant to our study, are presented. Video recording or user observation are examples of methods to gather data about user behavior/performance, while performing tasks. The use of sensors to collect user performance, and save this data in log files, is another method that can be used. Conducting a series of interviews with users, or asking them to do surveys, or sending them online surveys, are methods used to collect user feedback after the completion of the experiment. A concern to consider is the problem with using different methods to collect data. Unification of data gathered form different methods is a solution to that problem. The motivation is to be able to transform these different types of collected data into a single type of data, in order to more rapidly properly analyze and compare it with other collected data [13].

System-centric evaluation

System-centric evaluation is focused on evaluating ubiquitous environment's prototypes. This evaluation has also as focus to assess the developers themselves, i.e, assess the ability of the developers to identify unwanted behavior in the ubiquitous system.

Developer-centric evaluation This evaluation is mainly concerned with knowing if the developer can develop accurate ubiquitous environments. This can be accomplished by providing them with instructions on how to implement and configure a virtual environment, and then collect their feedback while performing a predefined prototyping task, in order to understand how easy it was for them to implement what was being proposed [26, 37]. In [17], a similar method is used to evaluate developers' performance. They are concerned with supporting interactive developers in the initial stage of the development. Also, a quick way to test and analyze their designs is through the use of storyboard analyses and recording/replay of test experiments. Other approach is to use developers as test users, in order to determine if they can identify problems in ubiquitous environment, and consequently identify their unwanted behavior.

Environments evaluation Evaluating environments is a crucial task for most developers/designers. Some ubiquitous systems have the goal to create virtual environments that are replicas of real environments. These virtual environments must have exactly the same properties that the real environments have, in order to give to the users a more immerse, and realistic experience. In order to assess these environments, users that regularly explore the real environment, should supply feedback to the developers, for them to know if the environment is reliable enough. Environments also need to be tested and evaluated at the level of modeling of virtual environments and its objects, in order to create the desired experience to use. It is necessary to check how the models perform without any specific interaction and how they react to user interactions or to context changes.

User-centric evaluation

User-centric evaluation focuses on how the users react to the ubiquitous system. Evaluating the users behavior and their feelings when interacting with it, or evaluating if they can interact with the system efficiently and perform the tasks they were assigned.

Evaluating user experience User experience can be characterized by how well a person feels, when she interacts with the system. One of the most important questions about user experience, that the developers want to answer, is "Did the user have a pleasant experience when interacted with the system?". However, this evaluation is not always reliable because it is subjective, i.e., it depends on the users' feelings when they are interacting with the system.

Through user experience evaluation, developers can know if the system that they are building will create a positive impact in people's lives. User experience evaluation techniques are widely used in many of the studies that were analyzed. Particularly in [17, 37, 13], a big importance is given to analyzing and comparing user experience and behavior, allowing them to eventually redefine the systems depending on the users' feedback.

Evaluating usability Usability relates to the ability of the user to use a certain object. In this case, the aim is to know if the ubiquitous system is easy for the user to use. According to the ISO 9241-11 standard [12], usability is a subset of user experience and it can answer the question, "Can users accomplish their goals?", having in consideration factors as satisfaction, efficiency and learnability. Usability is also a key goal on the process of developing ubiquitous systems. The more common approach to usability tests is done through observing and recording a user while he perform tasks. Others usability test methodologies are described in [22].

Maly et al. [13] built a framework for testing the usability of applications in virtual environments. The method consists on conducting specific tasks to evaluate usability. The approach builds on usability testing methodologies for desktop applications, combined with the evaluation of user behavior in ubiquitous environments.

3.3 Analysis

The relationships between the four dimensions of evaluation and the dimensions related to the development of ubiquitous systems is presented in Table 3.1. The table should be read having in consideration that the primary point of analysis are the several types of evaluation. Each evaluation dimension is analyzed towards each prototyping dimension, and also to the controlled experiments dimension. With this we want to highlight which dimensions are more critical for each evaluation dimension. The scale of values chosen to measure the relationship was: (1)- little influential, (2) - influential, and (3) - very influential. The values in Table 3.1 are derived from the analysis of the papers mentioned in this section. This values also emerged based on the percentage of codes collected for each of the dimensions of evaluation, when compared with the percentage of codes for each of the prototyping dimensions of the papers analyzed.

	Developer evaluation	Environments evaluation	Evaluating user experience	Evaluating usability
Fidelity of immersion	2	3	3	2
3D modeling and simulation	2	2	2	2
Embodied interaction support	2	1	3	3
Controlled environment manipulation	3	2	1	1
Context driven behavior	3	3	2	2
Multi-user support	2	2	1	1
Hybrid device prototyping	2	2	2	2
Controlled experiments	3	3	1	1

Table 3.1: Relation between each evaluation dimension and each prototyping dimension.

From Table 3.1 several conclusions can be reached. Developer-centric evaluation is more concerned with how to give behavior to the system, and how it reacts to change (be it context changes or user interactions). The ability to support multiple users with the purpose of realizing experiments is also an influential aspect of developer centric evaluation. Nevertheless, the other dimensions are also influential in this type of evaluation.

Regarding the assessment of environments, the more realistic is the environment, the better is the ability to evaluate the envisaged design. The realization of controlled experiments in the virtual environment, and how ubiquitous applications or smart objects react to changes are also among the most influential dimensions to assess environments. Allowing multiple users to interact with the environment, and supporting the use of virtual/real devices/services are the remaining influential dimensions in environment evaluation.

For the user to have a good user experience, he should feel able to use most of the interactions that he usually uses in reality. The environment should be as realistic as possible, so that the user feels as embedded as possible in the environment. Regarding usability, the way users interact with the system and how much they feel immersed in the virtual environment are the more important dimensions to make user more connected with the environment, thus providing them a better way to accomplish their tasks. Dimensions as the possibility of interaction with virtual or real devices/services, and the way the ubiquitous system reacts to the user, are other influential dimensions to usability evaluation.

3.4 Conclusion

Concluding, this chapter has presented a set of dimensions that address the prototyping of ubicomp systems and their evaluation. We believe that developers, before creating ubiquitous systems, should take into account some of the features that have been presented in here, for better planning the functionality of their systems. However, we are interested in ubicomp simulations platforms that create ubiquitous environments and that have concerns with evaluation. Therefore, the ubicomp simulation platforms mentioned in Section 3, UbiWorld [9], 3DSim [21], TATUS [25], VARU [11], APEX [37], are of particular relevance.

	UbiWorld	3DSim	TATUS	VARU	APEX
Prototyping dimensions	23	13	13	9	15
Evaluation dimension	17	5	7	4	10

Table 3.2: Relation between the codes of dimension and the simulation platforms.

In this context, Table 3.2 show the relation between the main groups of dimensions identified above and the ubiquitous simulation platforms. The codes of each dimension were grouped into the main groups of dimensions (prototyping and evaluation) for each of the platforms identified.

To summarize, based on Table 3.2 we can conclude that ubiquitous simulation platforms seem more interested in the dimensions used to characterize the prototyping of ubiquitous systems, thus seeming more concerned with the process of developing the prototypes, and with the improvement of the functionalities of the systems.

The fact that fewer codes were identified in the evaluation dimensions, can be explained by the fact that simulation platforms still give more priority to the prototyping process than to the evaluation of the prototypes. In what follows two prototypes will be developed with the goal of supporting evaluation of both the prototypes themselves and the systems being prototyped.

Chapter 4

Using the APEX Framework

In Chapter 2, frameworks for developing ubiquitous environments were presented. APEX is the framework which will be used for developing our case studies. This chapter is divided in two main sections. The first section, provides a more detailed description of the APEX framework. It describes the structure of its architecture, detailing each of its components. The second section presents the alternatives ways for users to experience the ubiquitous environments developed with the framework.

4.1 APEX Framework

The APEX framework [37] eases the iterative process of developing ubiquitous environments prototypes. This is achieved by connecting different components in a unified platform. The framework provides a library of virtual sensors (e.g. presence sensors and light sensors) and dynamic objects for modeling ubiquitous computing environments, and a software infrastructure for analyzing and animating the models.

4.1.1 Architecture

APEX was developed to provide a framework for rapid prototyping of ubiquitous environments. It is composed by four components: a virtual environment component, a behavioral component, a physical component and a communication/execution component. A global view of the APEX architecture is presented in Figure 4.1. Each of the components is described below.

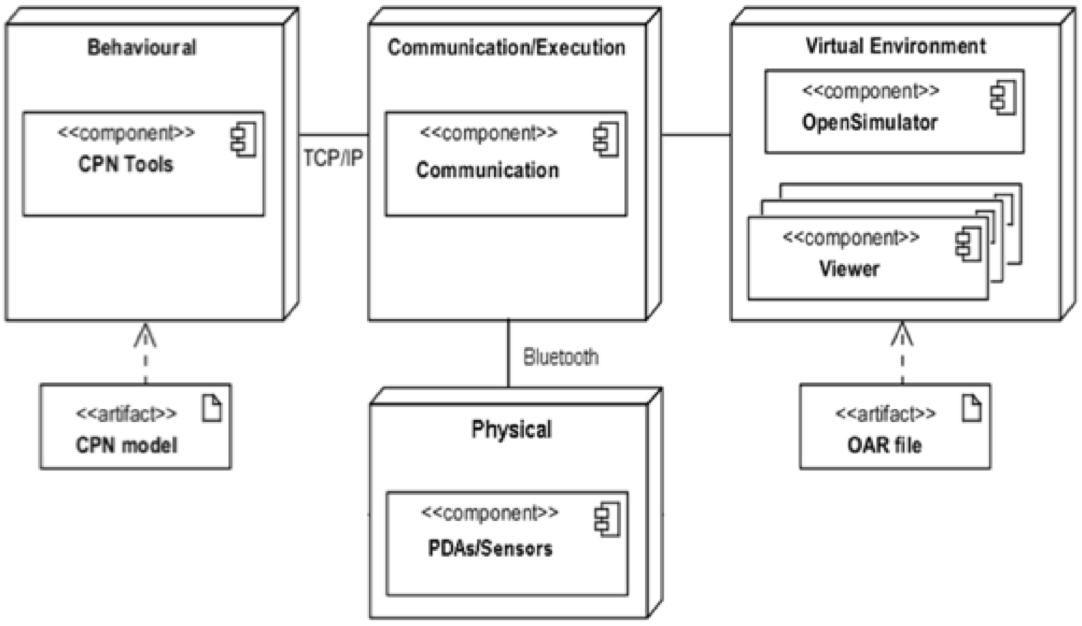


Figure 4.1: Logical architecture of the APEX framework

Virtual environment component This component is constituted by a OpenSimulator server and a viewer component which enables each user to connect to the server. OpenSimultator enables developers to create environments and customize them to best serve their interests. The OpenSimulator server also supports multiple users (in the same location or not) connected to the same virtual environment. Additionally it is responsible for controlling the possible interactions and the information in the virtual environment.

Virtual environments can be saved and loaded to different OpenSimulator servers via a specific file type, the OpenSimulator Archives (OAR). These files can be used to save and restore all that is associated with a virtual environment, i.e., the land, all objects, as well as its inventory (a database of objects available to be used in the environments). All the information associated with the objects (position, textures, animations, video and sound) is also saved in these files. A similar file type, the OpenSimulator Inventory ARchives (IAR) is specific to save only folders and items that exist in the inventory. Subsequently they can also be loaded into different OpenSimulator servers. It enables developers to save items, such as textures, scripts, objects or sounds that exists in a inventory, and subsequently restore them to enrich a different virtual environment.

The viewer component allows users to view the virtual environment (texture and physics of objects, and all the information about the environment) running on the OpenSimulator server. Furthermore, it allows users to interact with the environment by direct (with a device or a object) or indirect interaction (changes of context). The OpenSimulator server together with the client viewer permit to manipulate the virtual environment. This can go from manipulation of objects (changing the physics or texture of an object, e.g. lights), to the insertion/removal of sound, video or animations on objects. It is also possible to use scripts, by using the Linden Scripting Language (LSL), to give behavior to avatars and objects in the virtual environment. To do this, it is only necessary to select the object we want to give a specific behavior to, and associate a script to it.

It is necessary to have into account choosing a client viewer. The viewers most widely used and most appropriate to use are the Second Life viewer[1] and the Firestorm viewer[2]. A list of the alternatives viewers can be found in [1]. Some of the viewers can not fulfill some of the requirements of the developers, e.g, provide support for mesh objects, or the ability to interact with a local server.

Most of the communications made by the virtual environment component, is made with the behavioral component. The virtual environment component can send information related

[1]http://secondlife.com/support/downloads/ (Accessed: 29/1/2013)
[2]http://www.phoenixviewer.com/downloads.php (Accessed: 29/1/2013)

to a specific object/avatar to the behavioral component, such as the position of the avatar in the virtual environment. Or, the virtual environment component can receive the results of the decisions that were made in the behavioral component, and reflects them in the virtual environment (if an avatar is close enough to a gate, open it automatically).

Behavioral component This component uses CPN models to give behavior to objects in the virtual environment. These CPN models are created by CPN Tools. The models help developers to control and model the behavior of any object he desires to. For that to happen, the developer must insert the model of every dynamic object (devices, sensors) into a CPN base model. That CPN base model is used in APEX to provide assistance to modeling new virtual environment simulations. The base model consists of 3 types of modules:

- A module to establish the connection between the CPN model and the virtual environment server (OpenSimulator) and its devices' modules, and start the simulation;
- A module to receive data from the Opensimulator and update the relevant tokens in the model;
- Modules to describe the behavior of each device existing in the system

Using a CPN model to driven the behavior, enables CPN Tools to analyze systematically and exhaustively the behavior of a prototype. The State Space (SS) tool is integrated in CPN Tools and supports the verification of properties such as liveness or reachability, as well specific properties defined using the association programming language (CPN ML languages). This type of analysis falls outside the scope of this work, so it will not be discussed further. For more information see [34].

The communication of behavioral component is made primarily with the virtual environment component through the communication/execution component. Symmetrically to the communication made by the virtual environment component, the main communications made by the behavioral component are to receive the positions of objects and avatars, or any event associated with an object, and to trigger actions in the virtual environment, according to the conditions attached to such objects/events.

Figure 4.2 presents an example of a CPN module. This module describes the behavior of a gate with incorporated sensors, when a user approaches or moves away from it. The gate only has two states. The gate is **opened**, when a user is near the gate, and it is **closed** when no one is near it.

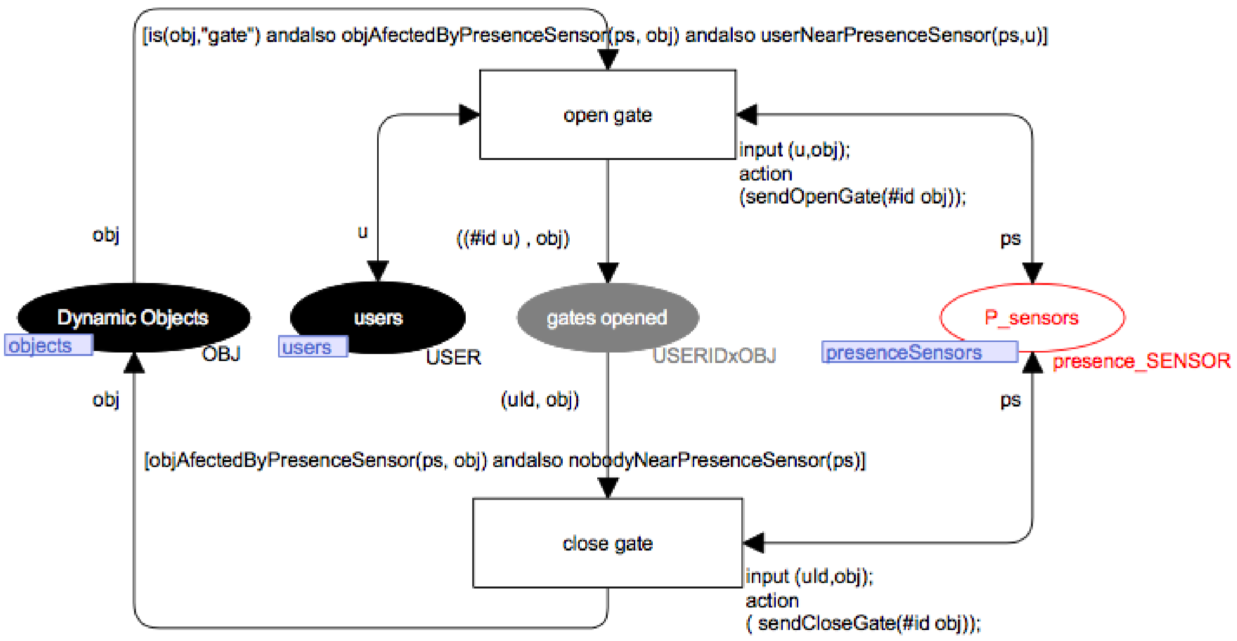


Figure 4.2: Gate module using a CPN model

Physical component This component is responsible for supporting the connection of real devices (smartphones, PDAs and sensors) to the APEX framework. This connection, established using Bluetooth, as can be seen in Figure 4.3, allows the framework to send and receive data (which can sometimes generate events in the virtual environment) from a real device. Besides the communication between the physical component and the virtual environment component, the physical component can also communicate with the other components. The communication between real devices and the other components (virtual environment and behavioral components) is established through the communication/execution component.

To use this feature a Bluetooth server application must be installed on each client machine, and a Bluetooth client application must be installed on the users' mobile device (currently Windows Mobile devices are supported). For the communication to be successful, a Bluetooth server must be selected and a user account must be provided in the mobile application. After this configuration, and after enabling Bluetooth on the mobile device, the APEX framework automatically detects the mobile device, and associates it with the previously set up user account.

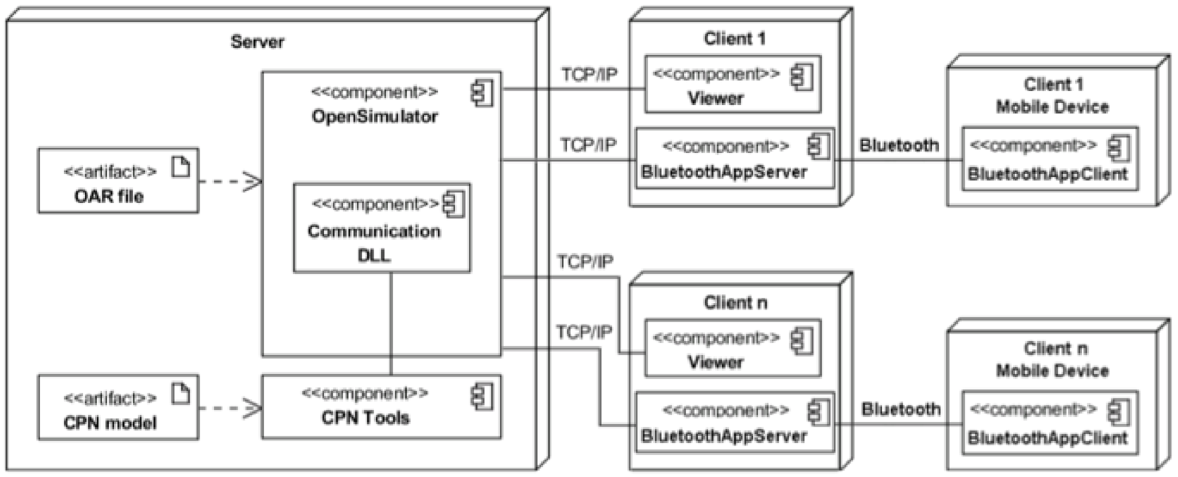


Figure 4.3: Physical Architecture

This feature can provide a better user experience. In general, this functionality gives users a better understanding of how the virtual environment works. First because users are immersed in the virtual environment. And second, because they experience the exchange of information between the real devices and the virtual environment, gaining a better understanding of how to interact with the environment and which events are generated from each interaction.

Communication/execution component This component acts like a communication bridge between all other components. Ensuring, that all communications are conducted appropriately, and that all the values changed are reflected almost immediately in the appropriated components. Thus, the communication/execution component maintains consistency between all other components. The component is implemented as a DLL implemented in C#. This DLL is loaded by OpenSimulator when it is initialized.

As previously stated, this component manages the communications made between the other components. The link between the dynamic objects in the virtual environment and its representations in the model, are only possible if two conditions are fulfilled. Firstly, it is necessary to specify the value of an unique identifier in all tokens in the CPN model that represent a dynamic object. Secondly, is necessary to put identifiers in the dynamic objects' properties, with the same values that were placed in the relevant tokens. Scripts must also be place within the dynamic objects in the virtual environment, for them to reflect changes according to their state in the CPN model.

Finally, the DLL allows specific commands to be invoked in the viewer of the OpenSimulator in real time. For example, to execute the loading/saving of a virtual environment, or to initializes the sensors that exist in the virtual environment (command INIT_SENSORS).

4.2 Second Life viewer

Appropriated tools should be provided to developers, in order to facilitate the process of building environments. Besides that, these tools should provide them means to develop richer content environments, and allow them to build virtual environments that may resemble a concrete, or envisaged, real environment. Creating virtual environments with rich content, in order to resemble real environments, is one of the ways for assessing the user experience in the final system. Such environments, should also provide users a better perception of the possible social interaction and ability to recognize difficulties regarding usability/interaction.

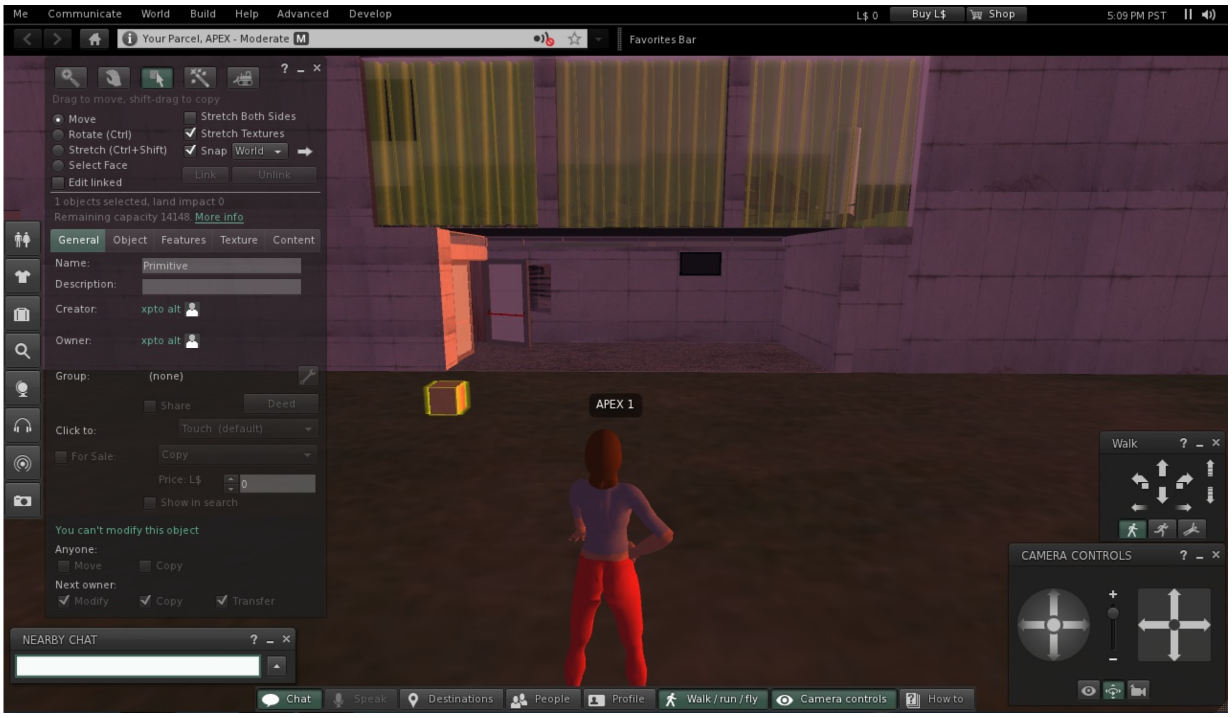
OpenSimulator can interact with several viewers (Second Life client viewers). Most Second Life viewers, provide the basic features to operate in 3D virtual environments (explore, interact, etc.), and providing tools to build those environments. But besides that, each viewer has specific features, that distinguish them from each other. For example, there are viewers to support specific visualizations (e.g. stereoscopic 3D visualization), viewers to operate with CAVE environments, and also viewers that target specific groups of users (closed communities). Below are presented the viewers which were used for developing this project. Some of the basic steps for constructing 3D virtual environments in Second Life/Second Life viewer are also described.

4.2.1 Client viewers

After Linden Lab released the source code of the Second Life official viewer, a wide variety of viewers began to appear[3,4]. Each viewer has specific characteristics to meet specific needs of certain groups of users. The Second Life viewer is appropriate for performing the most basic operations in the virtual environment only. These basic operations are, for example, to explore the virtual environment (walking around), interact with objects and with other avatars (talking to them). Customization of an avatar (changing its appearance and clothes) is also possible, and, if a user has enough privileges, he or she can also build environments/objects within the virtual world. With Second Life's official viewer it is possible to perform other tasks such as, buying items, earning money, among others. The complete list of features is provided in the tool's website[5]. Figure 4.4 shows an example of the Second Life viewer.

Two other popular viewers are the Firestorm viewer and the Phoenix viewer[6], both are

[3]http://wiki.secondlife.com/wiki/Third_Party_Viewer_Directory (Accessed: 29/1/2013)
[4]http://opensimulator.org/wiki/Compatible_Viewers (Accessed: 29/1/2013)
[5]http://wiki.secondlife.com/wiki/Features (Accessed: 29/1/2013)
[6]http://www.phoenixviewer.com/ (Accessed: 29/1/2013)


Me Communicate World Build Help Advanced Develop
L$ 0 Buy L$ Shop 5:09 PM PST
Your Parcel, APEX - Moderate
Favorites Bar
Drag to move, shift-drag to copy
Move
Rotate (Ctrl)
Stretch (Ctrl+Shift)
Select Face
Edit linked
Stretch Both Sides
Stretch Textures
Snap World
Link Unlink
1 objects selected, land impact 0
Remaining capacity 14148. More info
General Object Features Texture Content
You can't modify this object
Anyone: Move Copy
Next owner: Modify Copy Transfer
NEARBY CHAT
APEX 1
Walk
CAMERA CONTROLS
Chat Speak Destinations People Profile Walk / run / fly Camera controls How to

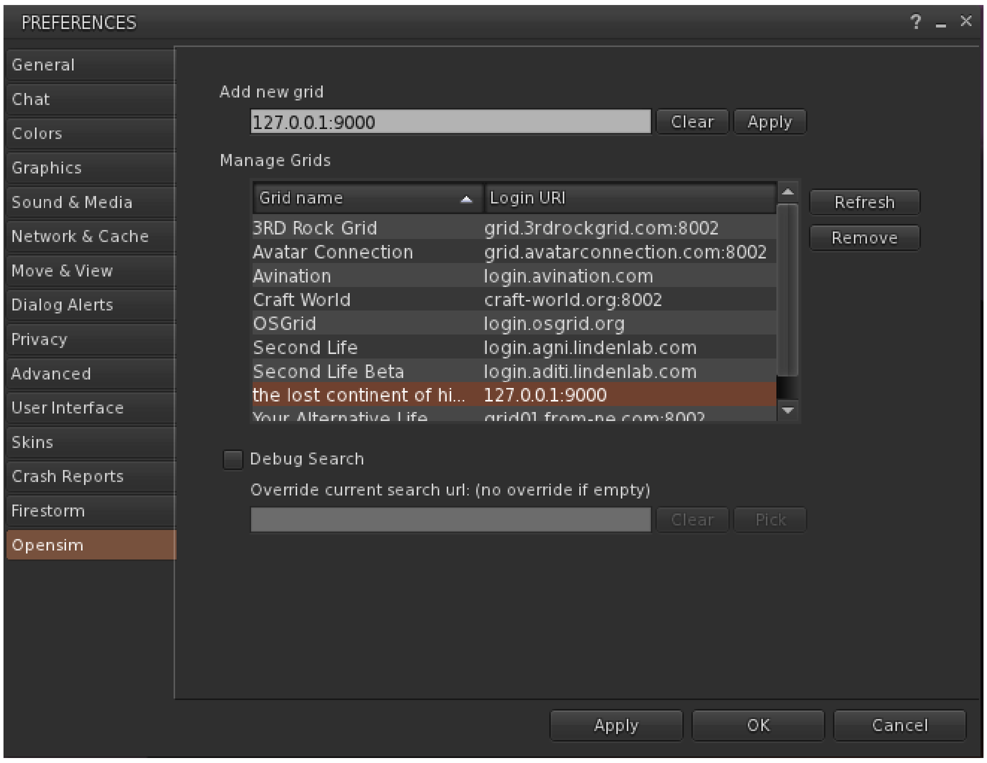
Name: Primitive
Description:
Creator: xpto alt
Owner: xpto alt
Group: (none)
Share Deed
Click to: Touch (default)
For Sale: Copy
Price: L$ 0
Show in search

Figure 4.4: Second life official viewer.

a community developed project provided by Firestorm Phoenix Project Incorporated. In specific, Firestorm viewer (currently the most used) aims to improve the user experience, by providing new features, and improving usability, functionality and flexibility of the viewer. It also aims to shorten the learning curve of users, to interact with a Second Life environment.

One of its most interesting features, and one of the main reasons for being highly used by the community that works with OpenSimulator, is the possibility to choose the OpenSim grid (i.e. the server) to which the viewer is connected. So far, available Second Life viewers are able to access OpenSimulator grids by defining their path when starting the server. This viewer is the only one that has the specific feature to access different OpenSimulator grids at runtime, as shown in Figure 4.5.

It is not always easy to build complex objects, starting with the basic objects supplied by Second Life. However, many viewers have the advantage to offer the ability to import



Add new grid

127.0.0.1:9000

Manage Grids

Grid name	Login URI
3RD Rock Grid	grid.3rdrockgrid.com:8002
Avatar Connection	grid.avatarconnection.com:8002
Avination	login.avination.com
Craft World	craft-world.org:8002
OSGrid	login.osgrid.org
Second Life	login.agni.lindenlab.com
Second Life Beta	login.aditi.lindenlab.com
the lost continent of hi...	127.0.0.1:9000
Your Alternative Life	grid01.from-ne.com:8002

Debug Search

Override current search url: (no override if empty)

Figure 4.5: Ability to connect to a specific grid.

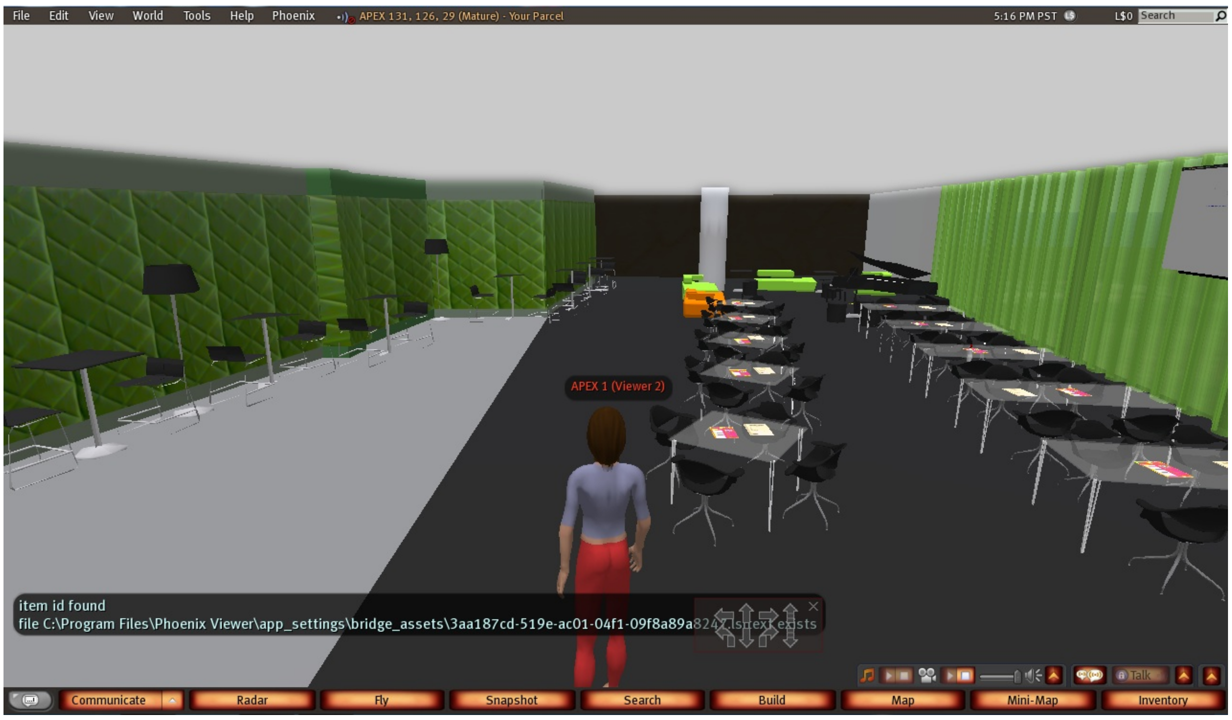
complex objects. There is some diversity in the type of objects that can be imported in Second Life, for example images, sounds or animations. The more complex objects (third party developed objects) can be imported in two ways. Some of the viewers like the Kokua or Imprudence viewers[7], allow importing XML objects. While most of viewers like the Firestorm viewer, Second Life viewer, among other allows importing mesh[8] objects.

Mesh objects can be built using third party tools, such as, Blender, Google Sketchup and 3DS Max. Their addition to a virtual environment, enables the development of more realistic, complex and accurate virtual environments, providing a better user experience. A huge amount of mesh objects is available for free in Google 3D Warehouse [9]. This is a repository with many 3D objects, like vehicles, furniture, everyday objects, or buildings.

However, not all combinations of Second Life viewers and OpenSimulator versions, support

[7]http://blog.kokuaviewer.org/ (Accessed: 29/1/2013)
[8]http://wiki.secondlife.com/wiki/Mesh/What_is_mesh%3F (Accessed: 29/1/2013)
[9]http://sketchup.google.com/3dwarehouse/ (Accessed: 29/1/2013)

the visualization of mesh objects. Figure 4.6 shows the use of the combination Phoenix viewer and 0.7.1 version of OpenSimulator, where is possible to visualize the mesh objects existing in the virtual environment. While Figure 4.7 shows the use of the combination of Firestorm viewer and the 0.7.1 version of OpenSimulator, and where it can be observed that it is not possible to show the mesh objects existing in the virtual environment (or they are shown with rendering problems).



```
item id found
file C:\Program Files\Phoenix Viewer\app_settings\bridge_assets\3aa187cd-519e-ac01-04f1-09f8a89a8247.lsltext exists
```

Figure 4.6: Environment showing mesh objects

Exist also other types of viewers with the property of making users feel more immersed in a virtual environment. For example, exist viewers for allowing the use for stereoscopic 3D visualization, or viewers to simulate or use in CAVE environments. Below, it is made a briefly description of each type of viewer. These viewers are already been used in the APEX framework and it is in line on one of the topics that has been earlier discussed in Fidelity of immersion.

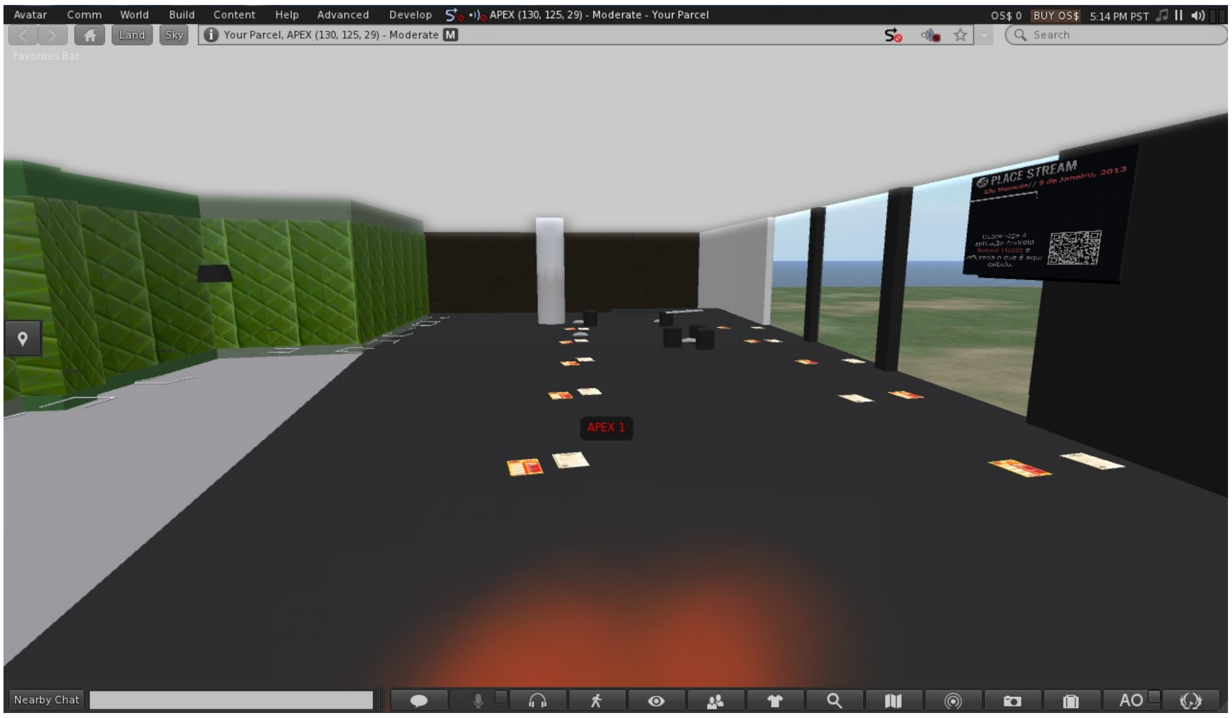

Avatar Comm World Build Content Help Advanced Develop
APEX (130, 125, 29) - Moderate - Your Parcel
OS$ 0 BUY OS$ 5:14 PM PST
Land Sky
Your Parcel, APEX (130, 125, 29) - Moderate
Search
Favorites Bar
PLACE STREAM
APEX 1
Nearby Chat
AO

Figure 4.7: Environment not showing mesh objects

The Dale's SL[10] is a viewer that improves the Second Life official viewer, by providing a feature to support to a stereoscopic 3D visualization. This feature provides to the user a better user experience, by improving user immersion in the virtual environment. This viewer provides 3 stereoscopic modes available to be used. In Dale's SL website[11] is provided more detailed information about stereoscopic modes. A short description is given below.

- In **Anaglyph Stereo** the sensation of depth is achieve through the use of red/cyan glasses. Figure 4.8 is an example of this mode;
- **Passive Stereo** is achieved through the use of two projectors, using polarized filters. This can also be achieve through the use of Head-Mounted Display (HMD). Passive Stereo is the mode which provides the most quality to the user;
- **Active Stereo** requires shutter glasses and one projector, for a user to view image

[10]http://sl.daleglass.net/ (Accessed: 29/1/2013)
[11]http://sl.daleglass.net/#stereo (Accessed: 29/1/2013)

depth. Stereo effect is achieved by separating the frames shown for each eye.

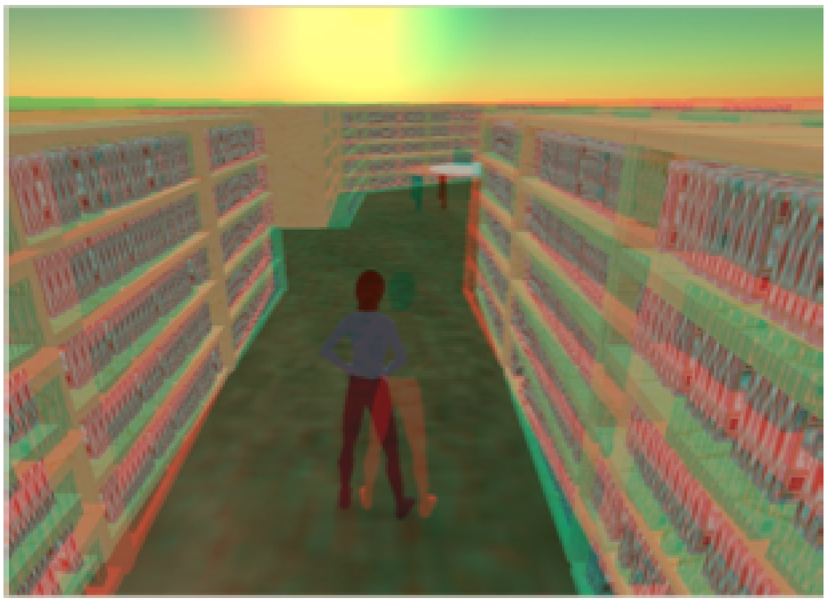


Figure 4.8: Example of Dale's SL viewer in anaglyph stereo mode.

CaveSL[12], as the name indicates, is a viewer for specific use in large scale immersive displays, such as a CAVE. One way to bypass the use of a CAVE (for monetary and logistic difficulties), is to have multiple monitors to show the virtual environment divided between them, as shown in Figure 4.9. The benefits of using CaveSL are:

- Support to use CAVEs and multi projectors;
- The ease to extend the number of displays used to show the virtual environment;
- Provides immersion, which improves the user experience;
- Solves most implementations difficulties between multiple displays, such as synchronization camera rotation, position and Field Of View.

More information about the possibility of integrating these features (stereoscopic 3D visualization, and multiple display support) with APEX are available in [19].

[12]http://projects.ict.usc.edu/force/cominghome/cavesl/index.html (Accessed: 29/1/2013)

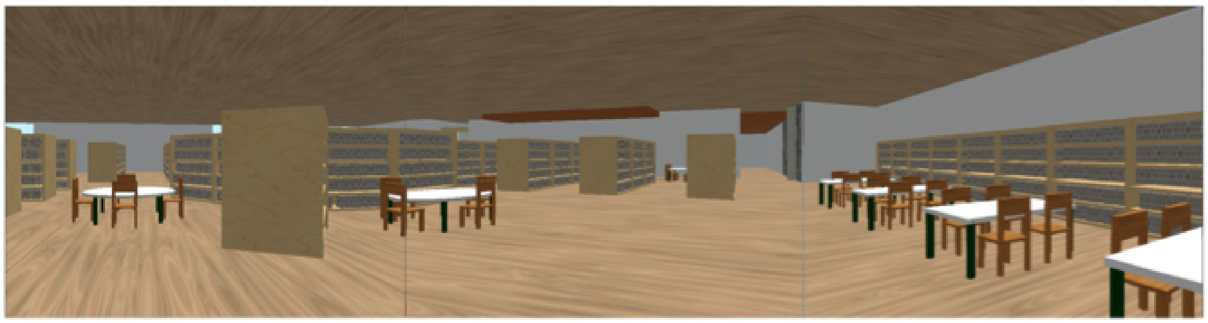


Figure 4.9: Example of CaveSL viewer with three running client viewers.

4.2.2 Second Life user interaction

The previous section presented several of the existing Second Life viewers, each oriented to specific features. Hereupon, using that viewers it becomes possible for users and developers to create a virtual environment, as well as to explore and interact with it.

The process starts by creating/adjusting the land where the desired environment will be built. Second Life viewers and OpenSimulator provides tools to create and adjust the land. Then, Second Life viewers provide a set of primitive objects (called ”Prim” objects), see Figure 4.10, that help developers build more complex objects (buildings, chairs, etc.), in order to enhance the virtual environment. These objects are customizable. Their size and position can be changed within the environment. Furthermore, the textures of these prim objects are also configurable. A set of options can be applied to objects to change their texture, e.g., the possibility to assign/change a color or even a more detailed texture.

One way to create more complex objects is achieved by linking together several primitive objects into a single and more complex object, e.g. ladders, chairs, as can be seen in Figure 4.11. Another way to incorporate complex objects in the virtual environment, is by importing mesh objects (see Figure 4.12). As already discussed, this option is only available in some Second Life viewers. Regarding the avatars, a few options are available to customize them, e.g. change their facial and clothing appearances.

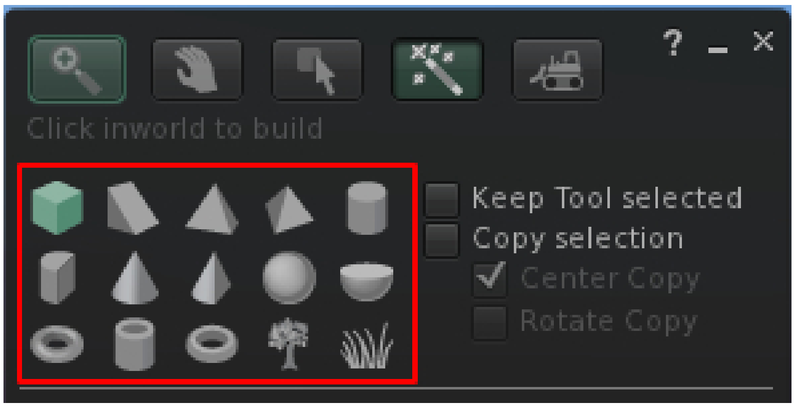


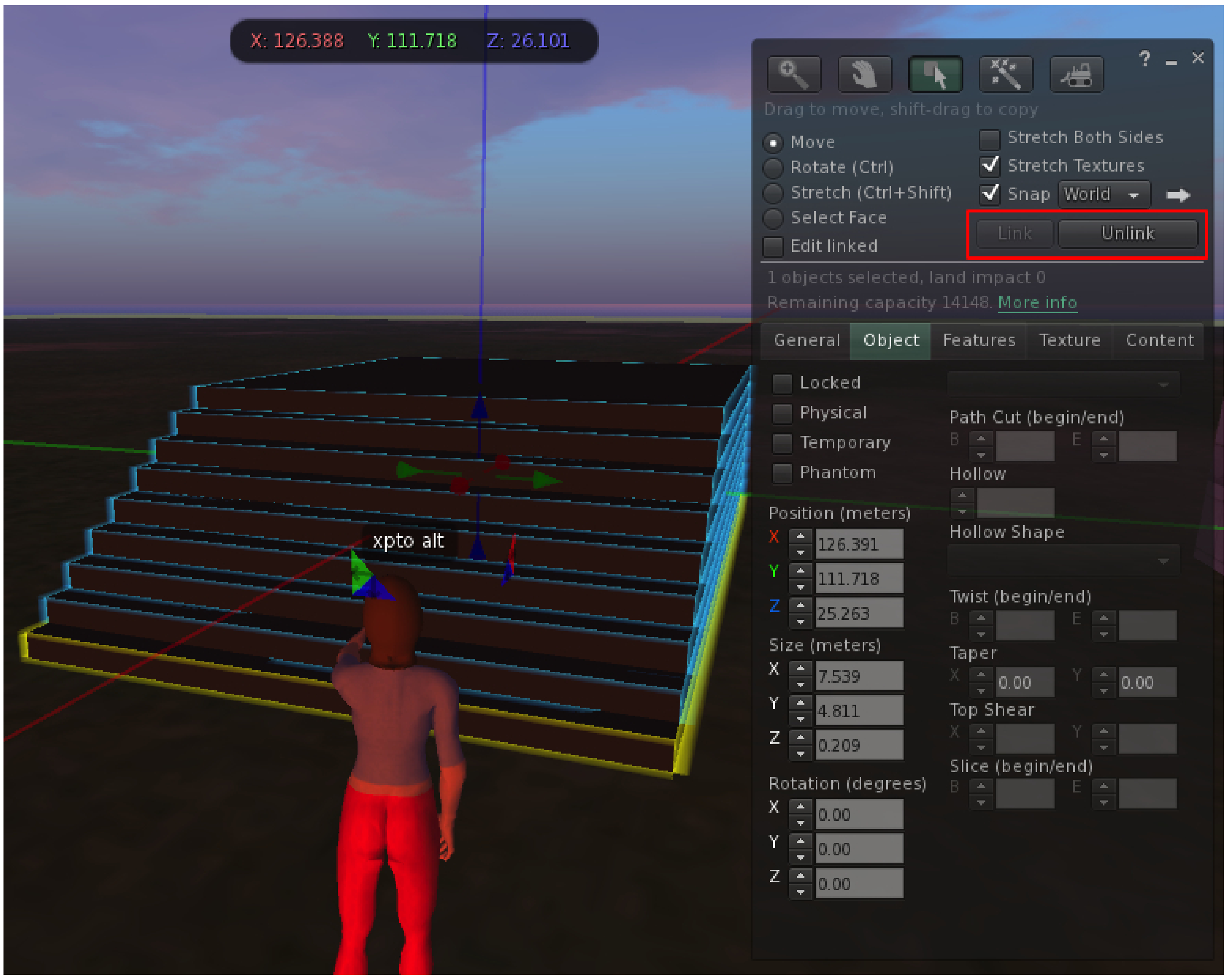
Figure 4.10: Primitive objects provided by Second Life viewers.

Adding new avatars to a virtual environment is only possible through the use of OpenSimulator. Exist two ways of adding new avatars to the virtual environment. The first is by using Non Player Characters (NPCs), which is possible by executing C# scripts in the OpenSimulator, or by executing specific commands in the OpenSimulator console. The other alternative, is having real users controlling avatars, and this is also achieved by executing a command at OpenSimulator console, that creates a new user account for a specific virtual environment.

4.3 Conclusion

The APEX framework supports rapid prototyping, making it easier and simpler to develop complex ubiquitous computing environments. APEX is composed by several components, each responsible for specific features (physical, behavior, simulation, communication). The connection between this components, permits that virtual objects and physical devices be available to the users enabling them to explore/interact with the virtual environment. Also, this separation of components allows exploring a design from a variety of perspectives. For example, using CPN models at the modeling component, virtual devices at the virtual component, and physical devices via Bluetooth at the physical component.

For improving the user experience of these virtual environments, several Second Life viewers



● Move
Rotate (Ctrl)
Stretch (Ctrl+Shift)
Select Face
Edit linked
Stretch Both Sides
✓ Stretch Textures
✓ Snap World
1 objects selected, land impact 0
Remaining capacity 14148. More info

Locked
Physical
Temporary
Phantom

Position (meters)
X 126.391
Y 111.718
Z 25.263

Size (meters)
X 7.539
Y 4.811
Z 0.209

Rotation (degrees)
X 0.00
Y 0.00
Z 0.00

Taper
X 0.00 Y 0.00

Figure 4.11: Linking primitive objects to build a more complex object.

can be used. Each supports specific features, such as CAVE environments or stereoscopic 3D. These viewers also allow developers to build virtual environments, by using resources provided by the same viewers (Prim objects), by OpenSimulator, or by importing mesh objects. This makes it possible to create virtual environments as close as possible to the target physical ubiquitous environments, providing users with a more complete and enhanced experience.

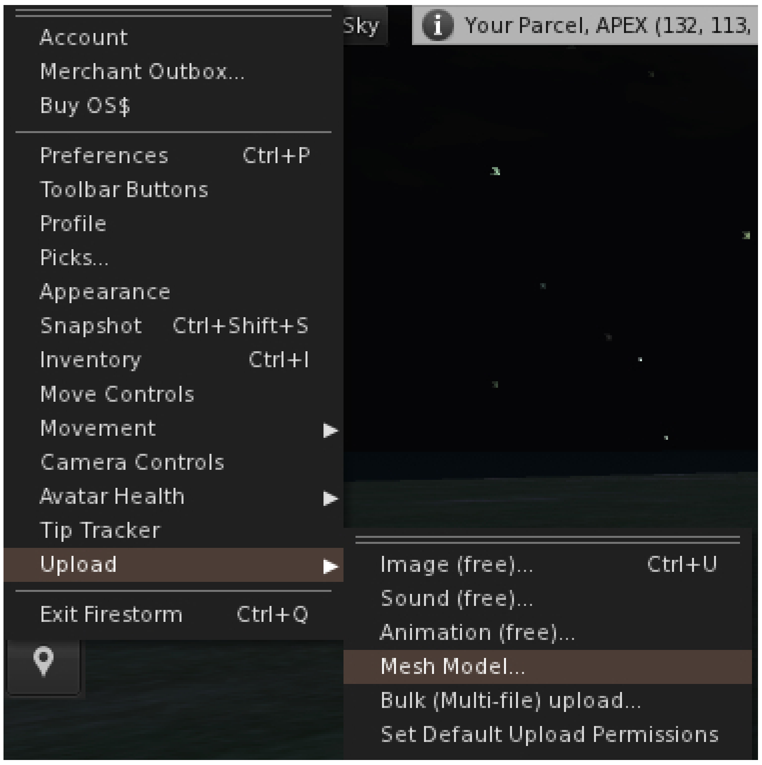


Figure 4.12: Functionality to import mesh objects to virtual environments.

Chapter 5

Case studies

This chapter presents two case studies implement with the APEX framework. For these case studies, two virtual environments were built with the purpose of assessing aspects related to virtual environments and also aspects of usability. While previous work had focused on the modeling layer [37, 34], and on the viewers [19], here the focus is on the the simulation, and in its connection to the physical world.

The first case study was develop to replicate, in a virtual environment, a users study previously carried out in real life. The second case study was developed to test ideas for an ongoing project of developing an ubiquitous system, which meant integrating the virtual environment with the software infrastructure already in place for that project.

5.1 First case study

The first virtual environment created was a recreation of the environment described in [40]. The purpose behind creating this first virtual environment had two reasons. First, it was used as an initial study of how to model ubiquitous computing environments in OpenSimulator. Secondly, it was used to support the assessment of whether the behavior of users inside a virtual environment will be similar to their behavior in the corresponding real environment.

5.1.1 The original experiment

The experience made by Varoudis et al. [40] consisted in analyzing if ambient displays, used as virtual extension of the limits of human vision in public spaces, influenced the visual relations between spaces and as consequence, changed people's movement.

The environment consisted on a corridor with a "T" shape. At the end of the corridor there was a wall separating the corridor from the next room, as can be see in Figure 5.1. After the wall, there was a coffee room where coffee was being offered (in order to encourage users to go through the corridor). In the corridor, there was a display broadcasting what was happening in the coffee room. The display was placed sometimes on the left, sometimes on the right, towards the end of the corridor. The purpose of this whole environment was to see if the position of the display (broadcasting what was happening in the coffee room) somehow influenced the decision making of the people going through the corridor (whether they turned the left or right when they arrive at the end of the corridor). A physical corridor and coffee room were built in order to carry out the experiment.

The results presented by Varoudis et al., show that a ambient display influences user's behavior (in the case, the route they took), when showing (broadcasting/real time stream) the place that the user wants to go. For example, one of the results obtained by Varoudis, indicates that placing the display on the left wall meant that the percentage of users who choose to turn left was 73.4%. And with the display placed on the right side, a percentage of 58.9% of test users choose to turn to the right.

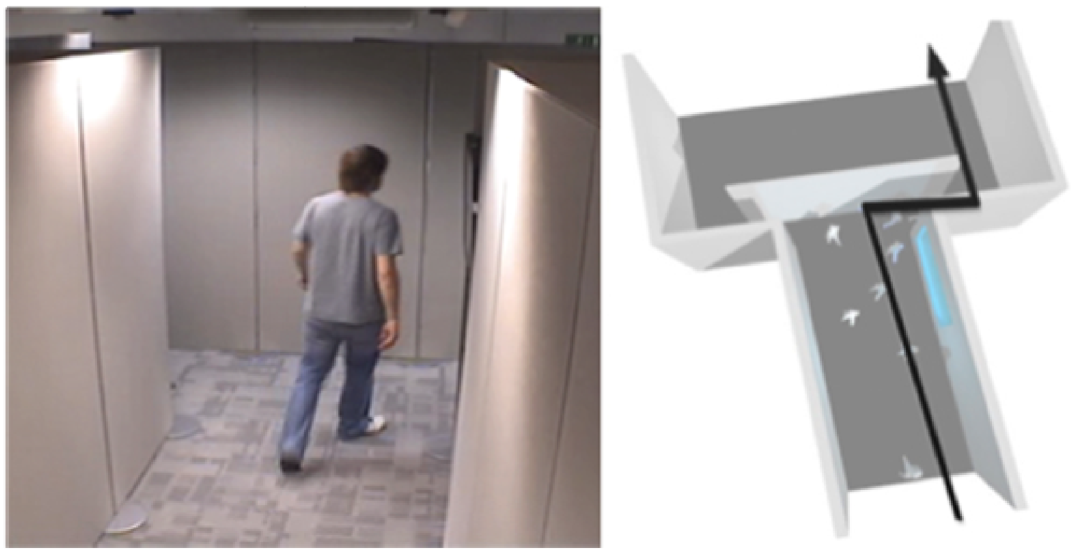


Figure 5.1: “T” shaped corridor with ambient display to the right (taken from [40])

5.1.2 Virtual environment proposed

In order to replicate the experiment, a virtual environment was created that is very similar to the environment in [40]. In our case, the 3D virtual environment was to be presented to and explored by post-graduate students, so instead of a coffee room, a faithful representation of the classroom where the students had lessons was constructed. Consequently, the display on the wall shows the classroom environment, not a coffee room. A further change is that, instead of having a live broadcast of what is happening in the classroom, the display shows the classroom environment by using a video. This change was made to simplify the implementation of the simulated display, since, given that short experiments were to be conducted, having a live feed was not deemed very relevant. The display was implemented by embedding a browser window in the environment.

As in real life, the virtual environment is composed by a corridor with the shape of a “T”, where the user must choose to turn right or to turn left in order to access a desired space. The environment is shown in Figure 5.2. As the figure shows, an ambient display is placed in one wall at the end of corridor (in this case, on the right).

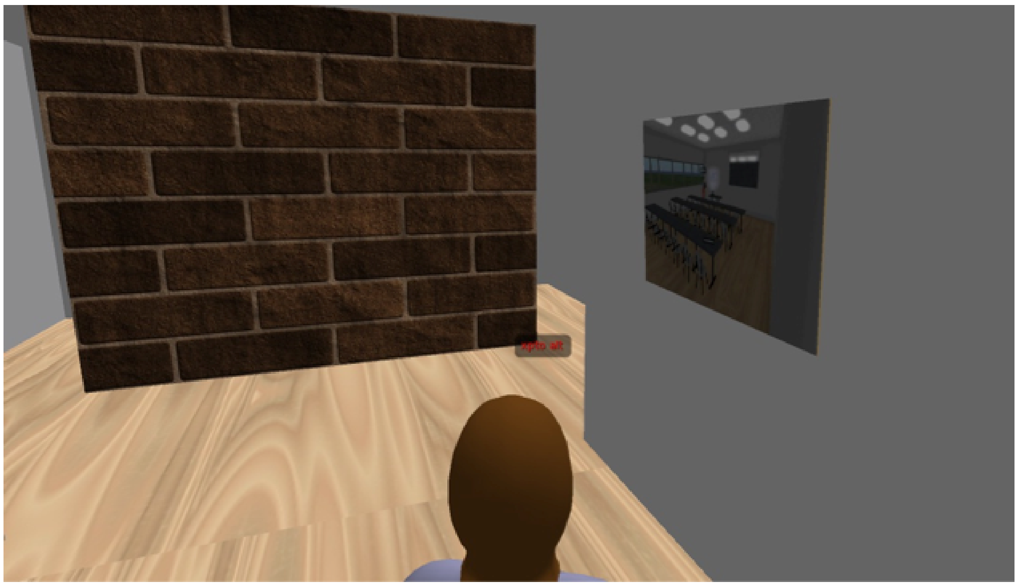


Figure 5.2: Example of experimental case

During the experiments with users, two conditions need to be tested. In the first, the display is placed in the right corner. In the second, it is placed in the left corner. The goal being to observe if, with the addition of an environment display, users' routes are changed. By analyzing and comparing the results from the experiments in the virtual environment with the results obtained by Varoudis et al., it becames possible to see if anything can be conclude about the users' behavior in the same experiments in "different" environments (real and virtual). A further goal is focused in analyzing the expressiveness of the APEX framework. In particular, if the navigation through the virtual environment seems realistic enough. Preliminary results of a study carried out with students from an Informatics doctoral programme indicate that indeed similar results are obtained in both the physical prototype and the virtual world prototype.

5.2 Second case study

The second virtual environment built, had the goal to represent as real and faithfully as possible the S. Mamede building in Guimarães. S. Mamede is a Arts and Shows Centre and it consists in a building with 3 floors. In the ground floor, there exists a shows room, where events are held (eg, theater plays, cinema, conferences, etc.), and also an art gallery that promotes exhibitions of art and photography. At the first floor there is a bar/restaurant, with a small area to host concerts or literary presentations. This is is a suitable area for social interaction. The second floor is composed by a library, and can also serve as a study room.

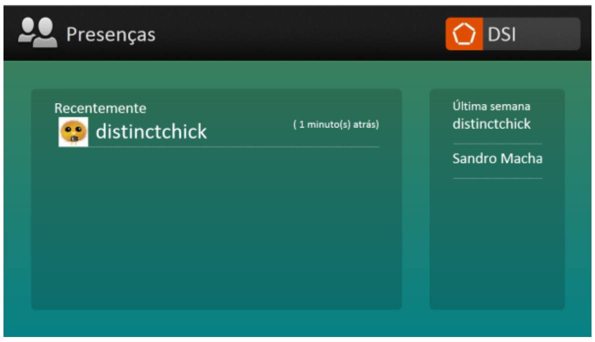
The first floor of S.Mamede features a system for interaction with public displays: Instant Places[1]. Instant Places supports interaction between users and displays using a mobile application, as it will be described below. The goal of this case study was to study extensions to the system. From a prototyping perspective, the main challenge was the integration of Instant Places with the virtual environment. Thus, we focused our work mainly on building the first floor of S. Mamede.

5.2.1 Instant Places

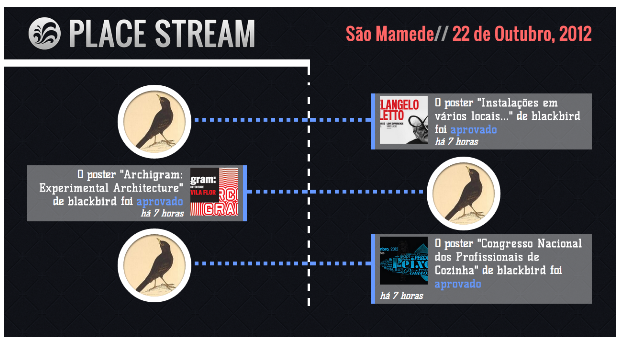
Instant Places is a system deployed in public displays, that makes it possible for a user to interact with the displays, and see new content constantly shown on them. This way, it allows new forms of expression in public displays. Namely, it allows any user to contribute with content to be displayed on the public displays. The public display is basically a display device connected to a network, able to interact and react to events in the environment. The system supports the interaction of a group of people present in the environment, and adapts the content shown on public displays, based on social situations and the preferences of the people are around it.

To interact with the public displays, it is necessary to have an account on the project's website. Upon creating and setting up the account, the user can interact with/send content to all the places that have public displays "running" Instant Places. Interaction with a public display it achieved through the Instant Places Android application, available on

[1]http://www.instantplaces.org/ (Accessed: 29/1/2013)



(a) Activity Stream



O poster "Instalações em vários locais..." de blackbird foi aprovado
há 7 horas

O poster "Archigram: Experimental Architecture" de blackbird foi aprovado
há 7 horas

O poster "Congresso Nacional dos Profissionais de Cozinha" de blackbird foi aprovado
há 7 horas

(b) Pins

(c) Posters

(d) Presences

Figure 5.3: Example of the four applications of Instant Places.

Google Play[2]. So far, the interactions that are possible to accomplish with the android application are to do a check-in and send posters to public displays. A user must be at a place where there is a public display to do the check-in. This interaction immediately influences the content that is shown in the display, based on the user preferences. The other interaction that is possible to accomplish is to share/recommend/send a poster to the public display, increasing the content that is shown on that display. It is also possible to collect into a smartphone the posters that are being displayed on the public display in a specific place, to later share those posters in other place.

At the moment, the content that is shown in the public displays with Instant Places, consists in four applications: Place Stream, Football Pins, Posters and Presence. Figure 5.3 shows examples of each of these applications:

[2]https://play.google.com/store/apps/details?id=instant.Places (Accessed: 29/1/2013)

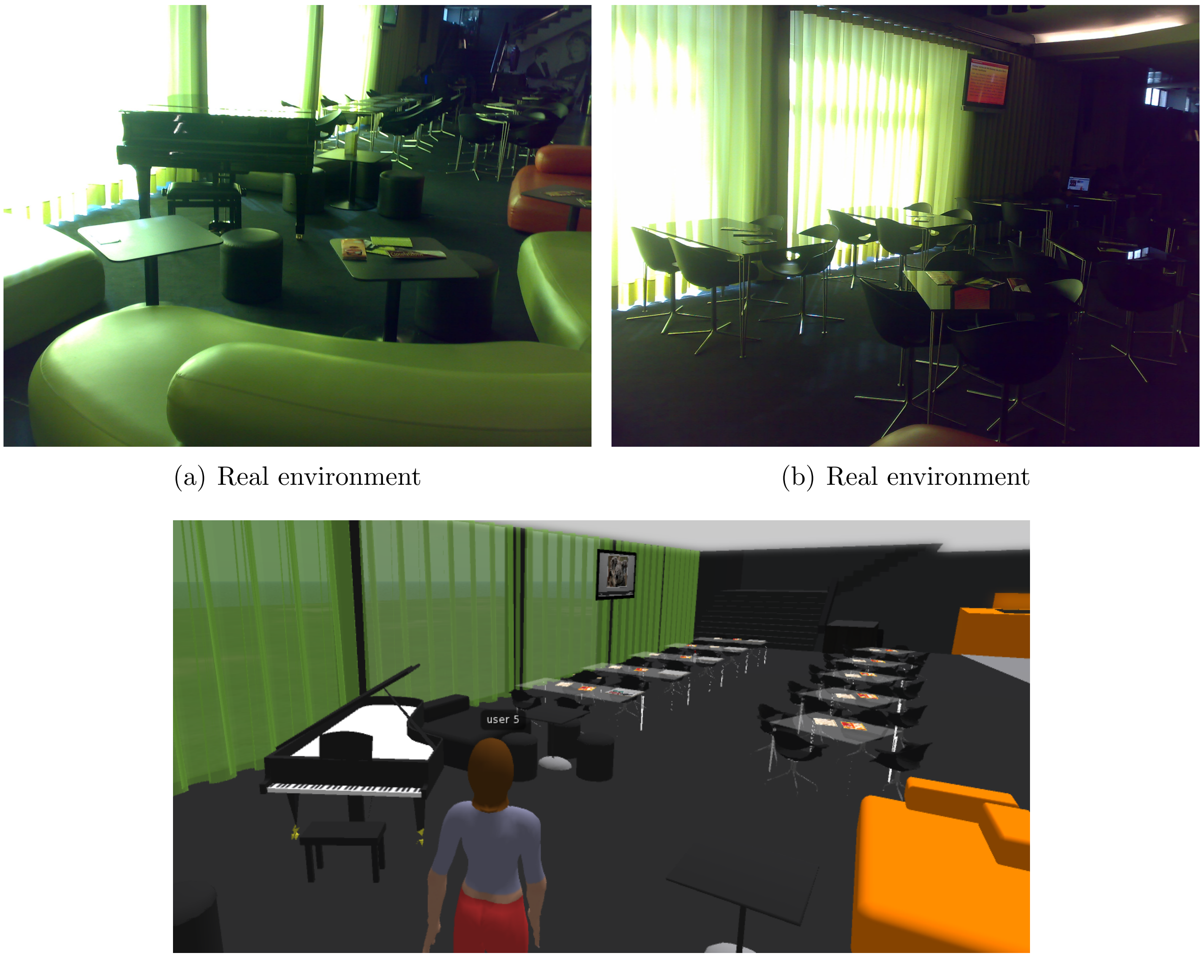
- **Place Stream**: Shows all the events triggered by Instant Places users for a specific place;
- **Football Pins**: Shows the football preferences of users that have checked-in in a specific place. In other words, the public display shows news/images of football clubs that users have as pins in their preferences;
- **Posters**: Shows the posters sent by Instant Places users. Users can send their poster to any place they visit (this can be used, for example, to promote an event);
- **Presences**: Shows a list of nicknames, and their pictures, of users that recently have made check-in in a specific place.

5.2.2 Proposed virtual environment

A virtual environment was built that tries to resemble the real environment as much as possible. All the physical characteristics of the real environment were implemented in the virtual environment, from the number of tables/chairs, to stairs, as well as every existing object in the environment. An example of the two environments (real and virtual) is shown in Figure 5.4.

Our proposal focuses on giving users the possibility to choose which Instant Places application they want to see in the screens inside the virtual environment. A new way to interact with Instant Places system was developed. Besides the already available use smartphones to choose which Instant Places applications to show on the public display, interaction through interactive (touch) tables was implemented (in the prototype). Testing this new interaction technology would be impossible in the real world scenario, due to the high costs associated with its implementation.

At the architectural level, since the Instant Places infrastructure is already assembled, it was not necessary to use the behavioral component of the APEX framework. All behavior will be obtained by integrating the Instant Places framework directly in the Virtual Envi-



(a) Real environment (b) Real environment

(c) Virtual environment

Figure 5.4: Example of the real and virtual environment.

ronment component. The architecture we envisaged is shown in Figure 5.5. Next, each of its components and how they interact with each other is explained in greater detail.

- **OpenSimulator**: Application server 3D used to "run" the developed virtual environment;
- **Website**: Represents the public display in the virtual environment. And is responsible for running the several Instant Places applications;
- **Web Service**: Responsible for manage which content (Instant Place application) shows in the public display in the virtual world;

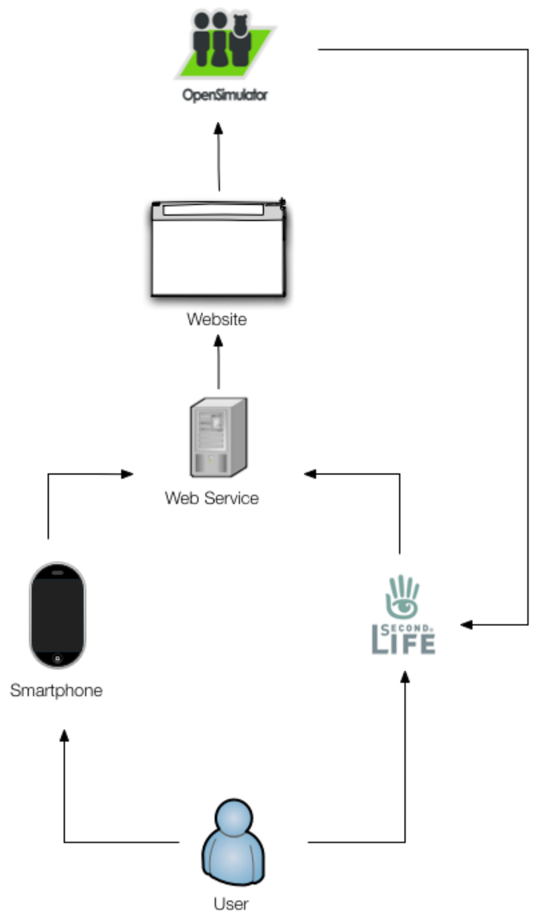
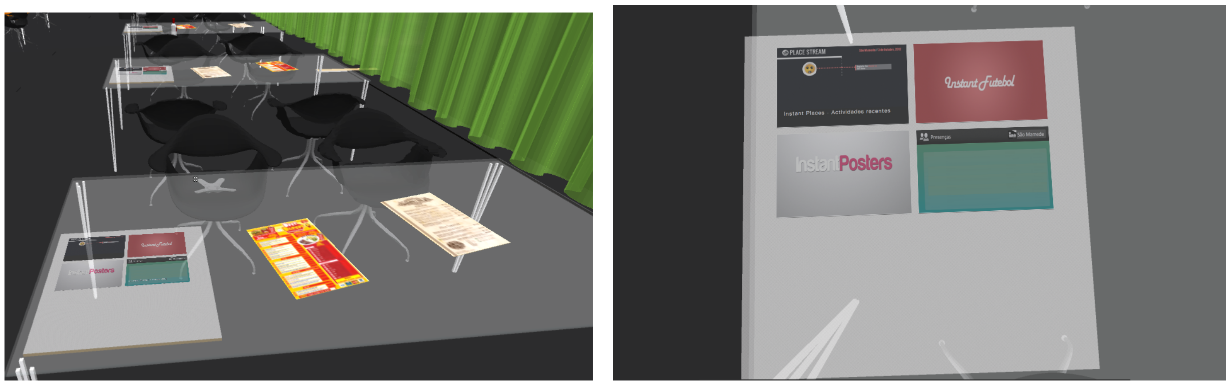


Figure 5.5: Representation of the architecture.

- **Smartphone**: A way that enables the users to choose which Instant Places application that they want to appear on the public display;
- **Second Life Viewer**: Second Life client that allows the user to interact with the virtual environment;
- **User**: User which will use the SecondLife viewer or the smartphone to choose which Instant Places application he wants to see.

Users interact with the Instant Place system, through the use interactive tables or through the smartphones. Interactive tables were constructed using a mixed between mesh objects and primitive objects. The table was used based on a mesh object. Additionally, primitive objects were placed on the tables. These objects are almost identical to the object created to simulate the public display in the previous section, but instead of passively showing a single page they show a set of applications to be selected, and support user interaction.

Each interactive table in the virtual environment shows all Instant Places applications available for the user choose from. An example, can be seen in Figure 5.6. When a user selects a Instant Places application from the interactive table, an action is triggered that sends a request to the web service, in order to change the application that is being displayed on the public display to the application that the user chose.



(a) Interactive Table (b) Display to choose applications

Figure 5.6: Example of the developed interactive tables.

The process for changing the application that is being displayed on the public display via a smartphone, is very similar to that of the interaction tables. All Instant Places applications displayed on the the public display are also available on the smartphone, as can be seen in Figure 5.7. Thus, when a user selects a specific application, a request is also sent to the web service, for changing the application that is being displayed on the public display inside the virtual environment.

The website was integrated into a browser and applied to a primitive object (Prim) of Second Life, in order to represent the public display in the virtual environment, as can be seen in Figure 5.8. The website is responsible for presenting the Instant Places applications to the user. It is constantly making requests to the web service, to verify if a user has made

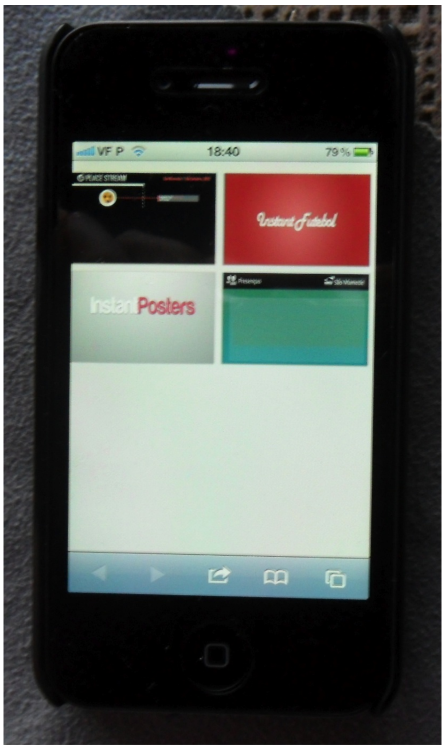


Figure 5.7: Method for choosing an application through a smartphone.

a request for changing the application that is being display in the public display. The web service receives requests from the users (through the interactive tables or smartphones) for the application that they want to see, and is responsible for managing requests for displayed application changes. The requests management logic can have several approaches. For example, it can give priority to certain requests (based on a counter of the number of requests per application). The web service also manages the time between application changes, and also notifies the users when they make requests. Several notification (pop-ups) can be sent to the user when he request an application:

- If there is no application to be shown in the public display, when a user request an application, a notification is shown to the user saying that the request was successful and the application will appear on the screen immediately;
- If there are other requests for applications to be shown in the public display, two things can happen:

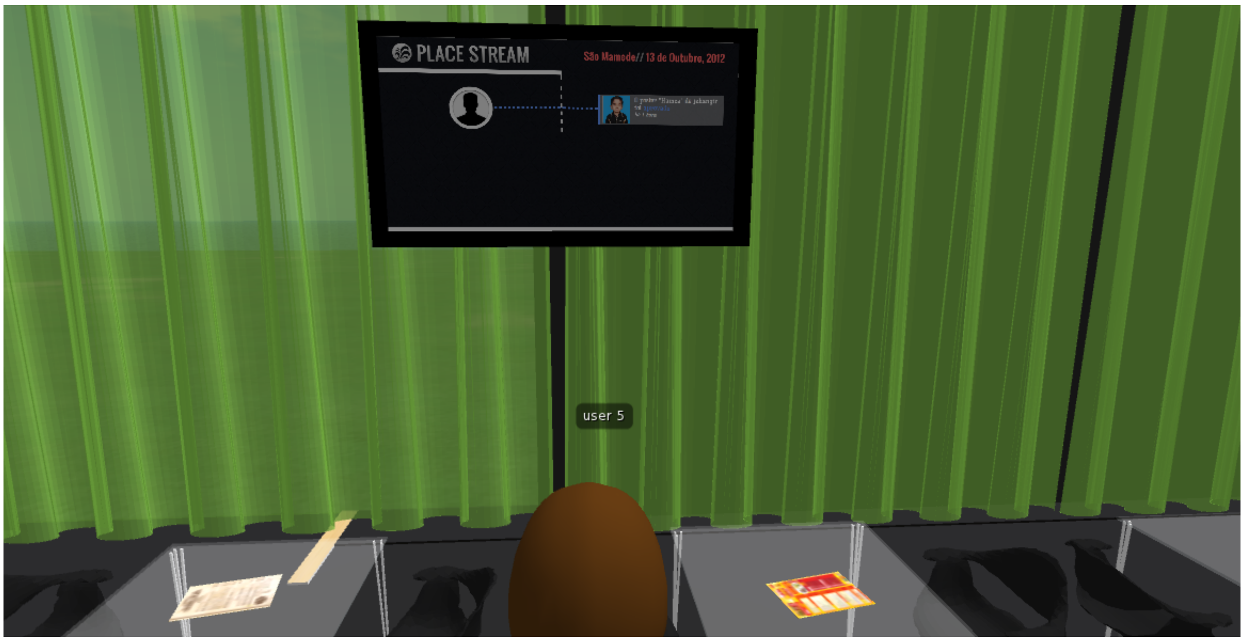


Figure 5.8: Public display showing Activity Stream application.

  – if the the application that the user requested is already in the queue of applications to be shown, the user receives a notification saying that there is already a request to display the requested application and it will be shown within x time;

  – if the application had no previous requests, then the user receives a notification saying that the request was successful, and it is also shown the waiting time until the requested application appears on the screen;

- Alternatively, if the request failed, the user receives a notification saying that it was not possible to fulfill his request.

Finally, the 3D applications server (OpenSimulator), is responsible for simulating the virtual environment. It simulates the entire environment (including the public display and all its content). Through the Second Life client users can explore the virtual environment.

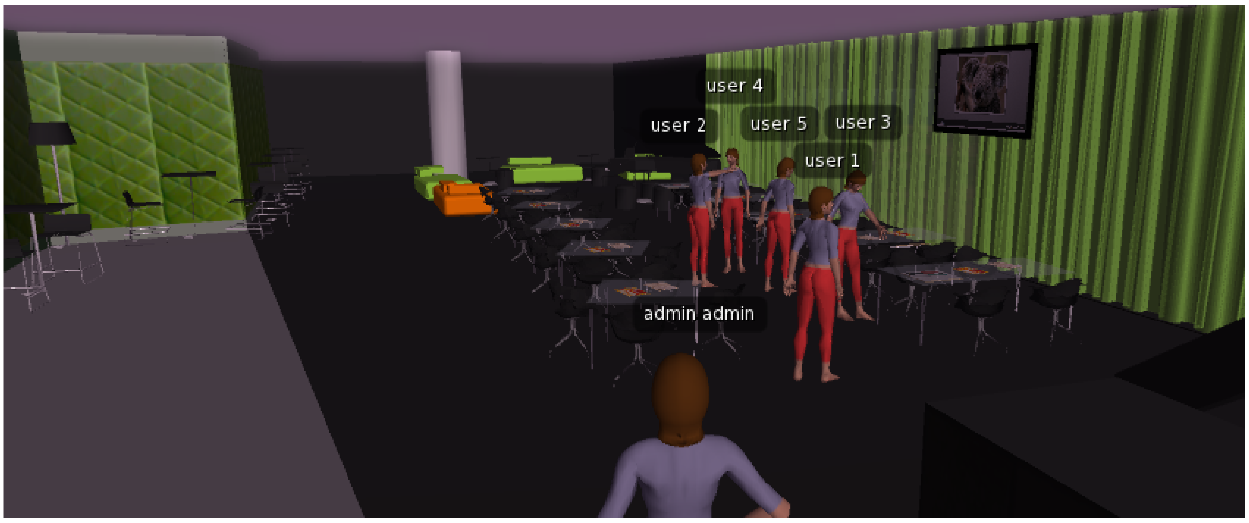


Figure 5.9: Test performed by users in the virtual environment.

5.2.3 Evaluation process and Results

With this prototype we want to evaluate specific situations. On the one hand we want to test the efficiency of the proposed system in supporting users perform certain tasks, and evaluate the users' experience while interacting with the system. On the other hand we want to evaluate how this proposed system behaves when multiple users interacts with it, requesting (simultaneously or not) the applications in the virtual environment and via smartphones.

Following this architecture and this system logic, were conducted two sessions of the same test. A total of 10 persons performed the test, and each test was realized by groups of 5 persons. Figure 5.9 illustrates users interacting with the virtual environment in one of the test sessions performed. More details about the test and about the persons who performed them are presented below:

- The age of the persons that carried out the tests were between 22 and 27 years;
- Most of the persons had an academic degree and had interest in technology;
- Each test was performed during 15 minutes;

- Users were told that they would explore a restaurant/leisure room. And that their goal was to try to influence what was shown on the public display;
- The system Instant Places system was explained to them;
- Users that had smartphones were given indications on how they could interact with the public display in the virtual environment through them;
- After users performed the test, they were asked to fill out two questionnaires (see Appendix A).

During the test performed by the users, the people responsible for monitoring the test were taking notes about user's feelings, about user's difficulties when interacting with the virtual environment, and also about possible errors in the development of the virtual environment.

The questionnaires and notes taken by the monitors had as main objectives to analyze user immersion in the virtual environment, and the usability of the Instant Places system in the virtual environment. We also wanted to analyze these new ways of interacting with Instant Places, more specifically whether the choice of applications is a feature that users find interesting.

Some interesting notes were taken by people who were in charge of monitoring the test. One of the notes concludes that only a few of the users used their smartphone to select which Instant Places applications they wanted to see in the public display. In fact only three persons used the smartphone. Another note states that the three people who had been at S. Mamede, recognized the virtual environment as the virtual representation of the S. Mamede environment.

As mentioned above, after the tests, users were asked to fill two questionnaires. One of the questionnaires used was the System Usability Scale (SUS)[3]. The SUS questionnaire is specific for evaluating usability. This is measured by several different usability aspects, such as the **effectiveness** and **efficiency** of the system and the user **satisfaction**. For

[3]http://www.measuringusability.com/sus.php (Accessed: 29/1/2013)

our specific case, with this questionnaire mainly we wanted to evaluate the usability of the new features implemented for the Instant Places system, and the usability of the system itself.

The SUS is a 10 item questionnaire with 5 different responses, ranging from strongly disagree to strongly agree. The SUS can be seen at Appendix A (first questionnaire). Scoring with SUS is subject to a few conditions:

- This scales all have values from 0 to 4 (with four being the most positive response).
- Specifically, for questions 1, 3, 5, 7, 9 the score contribution is:
  - Strongly Disagree = 0
  - Disagree = 1
  - Not sure = 2
  - Agree = 3
  - Strongly Agree = 4
- For questions 2, 4, 6, 8, 10 the score contribution is the opposite:
  - Strongly Disagree = 4
  - Disagree = 3
  - Not sure = 2
  - Agree = 1
  - Strongly Agree = 0
- The result of the test is calculated by adding up the converted responses for each user and multiply that total by 2.5. This converts the range of possible values to a scale from 0 to 100.

The average SUS score is 68, so any score above or around this value can be considered as good usability. After the analysis of the questionnaires filled by users, the average results obtained from them shows a score of 74. Thus, based on the results obtained, it can be stated that we obtained better results than the average results in the SUS questionnaires. Concluding, the the set of new features implemented for Instant Places (namely the combination of tabletop interfaces and smartphones), was considered as having good usability.

The other questionnaire was built based on the USE questionnaire[4]. In this second questionnaire (Appendix A - second questionnaire), we were more interested in evaluating mostly the usability aspects of the user being inside the virtual environment. We also made questions about the usability of the Instant Places system, but the majority of the questions concerned the user and the quality of the immersion provided by virtual environment. The questions in the questionnaire were grouped into three categories, to simplify the interpretation of the survey's results. The categories are:

- **System**: If the user understood how to interact with ubiquitous system;
- **Immersion**: If the virtual environment creates an immersive experience in the user;
- **User Satisfaction**: If the user is pleased after interacting with the virtual environment and ubiquitous system.

Interpretation of results was based on finding the mode (statistics) of each question of the survey and therefore of the categories created. In questionnaires that return ordinal data (agree, strongly agree), it is hard to tell the distances between the different scales. For example, the distance between neutral and agree may not be the same distance between agree and strongly agree. Therefore, in this type of questionnaire is recommended to use the (statistical) mode to interpret the results [32].

In the *System* and *User Satisfaction* categories a mode of 2 (agree) was obtained. In the *Immersion* category the bimodal 1 (somewhat agree) and 2 (agree) was obtained. While the number of test subjects is still small to enable statistically valid results, these results

[4]http://www.stcsig.org/usability/newsletter/0110_measuring_with_use.html (Accessed: 29/1/2013)

are nevertheless very promising. Both for the ability of APEX framework to create virtual environments and immerse users in it, and to provide a satisfactory experience to the user, and for the new approach to interacting in the Instant Places system (users easily understood how to interact with the ubiquitous system). However, the results enabled us to identify aspects of APEX that should be improved. Improvements at the level of immersion of the environment, which on one hand it can be enhanced by using the virtual environments inside a CAVE. And also some issues with users moving inside the virtual environments. Some issues were specific to the environment created, and others were because of how Second Life avatars move.

5.3 Conclusion

The virtual environments described in this chapter were based on studies already performed and ubiquitous systems already developed in the real world. The purpose was that of assessing aspects related to virtual environments and also aspects of usability. The results obtained through these initial tests were qualitatively good. The first environment had as initial objective to serve as study on how to model ubiquitous computing environments in OpenSimulator. Another objective was to assess if the behavior of users inside the virtual environment were in a certain way similar to their behavior in the corresponding real environment. A preliminary evaluation of the results help us to conclude that users usually take the same decisions both in the physical prototype and the virtual world prototype. And somehow also helped to strengthen the results obtained in the original experiment.

The second virtual environment was a faithful representation of the S.Mamede environment at Guimarães. It aimed to test the implementations of new ways to interact with the Instant Places system. Two questionnaires were conducted on this case, one more focused on the usability of the ubiquitous system, and the other more focused on the usability of the virtual environment and user satisfaction. Again the analysis of the preliminary results was positive. In the questionnaire addressing the ubiquitous system's usability, results indicate a score above the average for the results with that survey. This leads to a preliminary conclusion that the ubiquitous system and its new forms of interaction have good usability. Once the results are further validated, the next step is to inform developers of Instant Places about the results, in order to support their decision on the possible implementation of these new forms of interaction with theirs system in the real world. From the second questionnaire (more focused on the user and on the virtual environment), results were also obtained that indicate good usability and user satisfaction when interacting within the virtual environment. However aspects were also found that deserve consideration and improvement.

Chapter 6

Conclusion

6.1 Discussion

Ubiquitous systems present usability challenges in both design and development phases. User experience, in particular, is a difficult but crucial requirement which is difficult to measure, demonstrate and assess. The use of early prototypes of the envisaged system is a common approach to address this problem. However, given their situated nature, the development of such prototypes may imply design decisions and other associated costs, that will be very difficult to reverse. To address this, several ubicomp simulation platforms for the rapid prototyping of ubiquitous computing have emerged (see Chapter 2). These platforms offer different degrees of fidelity for the prototypes, from simple desktop simulations, to fully immersive experiences in a CAVE environment.

The APEX framework approach is one of the solutions that tries to solve the problems mentioned above. APEX offers a number of advantages in relation to other ubicomp simulators, for example multi-user support or exhaustive analysis support. Below, a list of features is presented:

- support for analysis through the simulation or through the CPN models (exhaustive analysis);

- multi-layered development approach supporting hybrid prototyping;
- multi-user support;
- focus on user's experience, and on how users will experience a virtual environment.

Other works have addressed the use of models for simulation and analysis [35, 37]. Here, the focus was on the integration of the virtual world simulations with actual *physical* services in order to create immersive hybrid prototypes.

To better understand what is involved in the prototyping of ubicomp environments, we need to establish how to align the prototypes with the key properties of the target environment, as well as the specific evaluation goals that they should support. Indeed, immersive prototyping requires thorough alignment with the key dimensions of the target environment, and a strong focus on the specific evaluation dimensions, such as users experience. Hence, in Chapter 3 we provided a framework to guide the alignment between specific evaluation goals and particular prototype properties (see Section 3.1).

This should provide a relevant contribution towards understanding the potential added-value of 3D simulation as a tool in the development process of ubiquitous computing environments. This is a proposal for developers consider before creating ubiquitous systems. They might take into consideration some of the dimensions that have been presented in here, to improve the planning and the functionality of their systems.

After the analysis that lead to the framework, two prototypes of virtual environments were implemented (see Section 5). These virtual environments were built to assess a number of aspects related mostly to the dimensions of Environments evaluation and User centric evaluation. By observing the behaviors of users inside virtual environments, and applying questionnaires, we were able to assess, both aspects related to the systems being prototyped and their influence on users, and to the prototyping themselves.

After this research, it were implemented two prototypes of virtual environments using

APEX. Some problems were detected using APEX framework. Starting with the compatibility of OpenSimulator versions using the DLL to run CPN models. And also with problems between OpenSimulator versions and Second Life viewers when using mesh objects to fill with content the virtual environments. Virtual environments were built basically to assess certain behaviors of users inside virtual environments.

The main goal of the first virtual environment was to support a study aimed at understanding if users have the same or similar behaviors and decision making procedures when faced with similar situations in both the physical and virtual words. An immersive prototype was created that simulated the same conditions used in a study where a actual physical environment had to be built. Although more tests would help to strengthen the results obtained, a posterior user study indicates that users "inside" virtual environments make the same or similar decisions to those they make in real environments.

The second virtual environment recreated an actual ubiquitous system which already operates in the real world. The environment recreated conditions that were not possible to easily test in the physical system (the introduction of interactive tables user interfaces) in order to carry out tests of new functionalities. On the one hand, the aim was to analyze the usability of these new functionalities, on the other hand it was also to assess the user experience of interacting with the virtual environment. The feedback obtained through the notes taken by monitors and through the questionnaires, in general, helped us to conclude that the overall ubiquitous environment had a good usability. More specifically, with the SUS questionnaire, which had a strong focus on the new functionalities of the ubiquitous system, we obtained better results than the average results of SUS questionnaires. This indicates that the ubiquitous system, as experienced through the virtual environment, has good usability aspects. The second questionnaire was more focused on assessing three categories: a) user satisfaction; b) quality of immersion of the virtual environment; c) and user perception of the ubiquitous system. Our results show that good results were obtained on all three categories. This indicates that, overall, the environment provided a satisfactory experience to the user.

Looking back at the two prototypes, it can be said that the prototypes covered most of the dimensions identified in Section 3.1, with an emphasis on Fidelity of immersion, 3D modeling and simulation, Multi-user support and Hybrid prototyping.

During the development of the prototypes, some problems were faced using APEX. Starting with the compatibility of OpenSimulator versions with the DLL implementing the Communication component, to with problems between OpenSimulator versions and Second Life viewers when using mesh objects to fill the virtual environments with content. to To solve this, appropriate versions of OpenSimulator and Second Life viewers were identified, in order to obtain the best use, offered by both these tools.

6.2 Current and Future Work

The work described in this dissertation was developed in the context of the APEX project. With the virtual environments developed, additional work is ongoing or planned. Work is being developed for the second case study (by the author), and it is also planned that more tests will to carry out on the first one (in the context of the project). To capitalize on the experience gained, the implementation of a new, larger, case study is also starting.

Summarizing, ongoing and planned work follows three main paths:

- New user tests are being conducted on the second case study. This new tests are being performed both on the virtual and the real environments. In this new experiment the logic of application selection in Instant Places has changed. Instead of the applications being displayed immediately after their selection, in this experiment users have a period of 4 minute to vote on the application that they want to see. At the end of that time, a ranking of the votes is shown and then the applications are then displayed according to the ranking. It is expected that greater interaction between users will happen, in order to work together to choose the applications.
- In order to consolidate the results obtained in the first case study is it expected that

more user tests will be performed.

- Another virtual environment will be built, to serve as another case study. This will be a representation of an existing nursing home that exists in Braga. While previous work on the APEX framework has focused mostly on exploring each of the components in the framework, this virtual environment will be used to explore the interaction between the components. Factors to be evaluated are still under study.

Bibliography

[1] Opensimulator compatible viewers. `http://opensimulator.org/wiki/Connecting`. Accessed: 29/1/2013.

[2] Usability evaluation by query techniques. `http://www.it.bton.ac.uk/staff/rng/teaching/notes/UsabilityEvalQuery.html`. Accessed: 29/1/2013.

[3] Usability testing. `http://www.usability.gov/pdfs/chapter18.pdf`. Accessed: 29/1/2013.

[4] Ali Asghar Nazari Shirehjini, Hannes Guddat, S. N. N. S. Platform for distributed multimedia environments supporting arbitrarily nested team structures. In *Proceedings of SPIE Vol. 5241 Multimedia Systems and Applications VI, edited by Andrew G. Tescher, Bhaskaran Vasudev, V. Michael Bove, Jr., Ajay Divakaran* (Bellinghm, WA, 2003), SPIE Society of Photo-Optical Instrumentation Engineering, pp. 169–179.

[5] Barton, J. J., and Vijayaraghavan, V. UBIWISE, A Ubiquitous Wireless Infrastructure Simulation Environment. Technical Report HPL-2002-303, HP Laboratories, Palo Alto, October 2002.

[6] Barton, J. J., and Vijayaraghavan, V. UBIWISE, a simulator for ubiquitous computing systems design. Technical Report HPL-2003-93, HP Laboratories, Palo Alto, April 2003.

[7] Cruz-Neira, C., Sandin, D. J., and DeFanti, T. A. Surround-screen projection-based virtual reality: the design and implementation of the cave. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1993), SIGGRAPH '93, ACM, pp. 135–142.

[8] Czernuszenko, M., Pape, D., Sandin, D., DeFanti, T., Dawe, G. L., and Brown, M. D. The immersadesk and infinity wall projection-based virtual reality displays. *SIGGRAPH Comput. Graph. 31*, 2 (May 1997), 46–49.

[9] Disz, T., Papka, M. E., and Stevens, R. Ubiworld: An environment integrating virtual reality, supercomputing, and design. In *Heterogeneous Computing Workshop* (1997), pp. 46–57.

[10] Hampson, C. *The Visualisation of Adaptive Behaviour in Ubiquitous Computing Experiments.* PhD thesis, University of Dublin, Trinity College, 2006.

[11] Irawati, S., Ahn, S., Kim, J., and Ko, H. VARU Framework: Enabling Rapid Prototyping of VR, AR and Ubiquitous Applications. In *Virtual Reality Conference, 2008. VR'08. IEEE* (2008), pp. 201–208.

[12] ISO, Ed. *ISO 9241-11: Ergonomic requirements for office work with visual display terminals (VDTs) – Part 9: Requirements for non-keyboard input devices.* 2000.

[13] Ivo Maly, Jan Curin, P. S., and Kleindienst, J. *Framework for Visual Analysis of User Behaviour in Ambient Intelligence Environment.* InTech, 2010.

[14] Jensen, K., Kristensen, L. M., and Wells, L. Coloured petri nets and cpn tools for modelling and validation of concurrent systems. *Int. J. Softw. Tools Technol. Transf. 9* (May 2007), 213–254.

[15] Kelley, J. F. An empirical methodology for writing user-friendly natural language computer applications. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems* (New York, NY, USA, 1983), CHI '83, ACM, pp. 193–196.

[16] Li, Y., and et al. Rapid prototyping tools for context-aware applications, 2005.

[17] Li, Y., Hong, J. I., and Landay, J. A. Topiary: a tool for prototyping location-enhanced applications. In *Proceedings of the 17th annual ACM symposium on User interface software and technology* (New York, NY, USA, 2004), UIST '04, ACM, pp. 217–226.

[18] Martin, M., and Nurmi, P. A generic large scale simulator for ubiquitous computing. *Mobile and Ubiquitous Systems, Annual International Conference on 0* (2006), 1–3.

[19] Moreira, R. *Master Thesis: Integrating a 3D application server with a CAVE.* PhD thesis, University of Minho, 2011.

[20] Nakanishi, Y. Virtual prototyping using miniature model and visualization for interactive public displays. In *Proceedings of the Designing Interactive Systems Conference* (New York, NY, USA, 2012), DIS '12, ACM, pp. 458–467.

[21] Nazari Shirehjini, A. A., and Klar, F. 3dsim: rapid prototyping ambient intelligence. In *Proceedings of the 2005 joint conference on Smart objects and ambient intelligence: innovative context-aware services: usages and technologies* (New York, NY, USA, 2005), sOc-EUSAI '05, ACM, pp. 303–307.

[22] Nielsen, J. *Usability Engineering.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[23] Nishikawa, H., Yamamoto, S., Tamai, M., Nishigaki, K., Kitani, T., Shibata, N., Yasumoto, K., and Ito, M. *UbiREAL: Realistic Smartspace Simulator for Systematic Testing.* 2006.

[24] ONeill, E. *TATUS A Ubiquitous Computing Simulator.* PhD thesis, University of Dublin, Trinity College, 2005.

[25] O'Neill, E., Klepal, M., Lewis, D., O'Donnell, T., O'Sullivan, D., and Pesch, D. A testbed for evaluating human interaction with ubiquitous computing environments. In *Proceedings of the First International Conference on Testbeds and*

Research Infrastructures for the DEvelopment of NeTworks and COMmunities (Washington, DC, USA, 2005), TRIDENTCOM '05, IEEE Computer Society, pp. 60–69.

[26] O'Neill, E., Lewis, D., and Conlan, O. A simulation-based approach to highly iterative prototyping of ubiquitous computing systems. In *Proceedings of the 2nd International Conference on Simulation Tools and Techniques* (ICST, Brussels, Belgium, Belgium, 2009), Simutools '09, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), pp. 56:1–56:10.

[27] Ostkamp, M., Bauer, G., and Kray, C. Visual highlighting on public displays. In *Proceedings of the 2012 International Symposium on Pervasive Displays* (New York, NY, USA, 2012), PerDis '12, ACM, pp. 2:1–2:6.

[28] Petri, C. A., and Reisig, W. Petri net. *Scholarpedia 3*, 4 (2008), 6477.

[29] Prendinger, H., Brandherm, B., and Ullrich, S. A simulation framework for sensor-based systems in second life. *Presence: Teleoper. Virtual Environ. 18*, 6 (Dec. 2009), 468–477.

[30] Reilly, D., Dearman, D., Welsman-Dinelle, M., and Inkpen, K. Evaluating early prototypes in context: Trade-offs, challenges, and successes. *IEEE Pervasive Computing 4*, 4 (October 2005), 42–50.

[31] Reynolds, V., Cahill, V., and Senart, A. Requirements for an ubiquitous computing simulation and emulation environment. In *Proceedings of the first international conference on Integrated internet ad hoc and sensor networks* (New York, NY, USA, 2006), InterSense '06, ACM.

[32] Robertson, J. Likert-type scales, statistical methods, and effect sizes. *Commun. ACM 55*, 5 (May 2012), 6–7.

[33] Rogers, Y. Moving on from Weiser's Vision of Calm Computing: Engaging UbiComp Experiences. In *UbiComp 2006: Ubiquitous Computing*, P. Dourish and A. Friday,

Eds., vol. 4206 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, Berlin, Heidelberg, 2006, ch. 24, pp. 404–421.

[34] Silva, J., Campos, J., and Harrison, M. Formal analysis of ubiquitous computing environments through the apex framework. In *ACM Symposium on Engineering Interactive Computing Systems (EICS2012)* (2012), ACM, pp. 131–140.

[35] Silva, J. L. *Master Thesis: Rapid Prototyping of Ubiquitous Computing Environments*. PhD thesis, University of Minho, 2012.

[36] Silva, J. L., Campos, J. C., and Harrison, M. D. An infrastructure for experience centered agile prototyping of ambient intelligence. In *Proceedings of the 1st ACM SIGCHI symposium on Engineering interactive computing systems* (New York, NY, USA, 2009), EICS '09, ACM, pp. 79–84.

[37] Silva, J. L., Ribeiro, O. R., Fernandes, J. a. M., Campos, J. C., and Harrison, M. D. The apex framework: prototyping of ubiquitous environments based on petri nets. In *Proceedings of the Third international conference on Human-centred software engineering* (Berlin, Heidelberg, 2010), HCSE'10, Springer-Verlag, pp. 6–21.

[38] Singh, P., Ha, H. N., Olivier, P., Kray, C., Kuang, Z., Guo, A. W., Blythe, P., and James, P. Rapid prototyping and evaluation of intelligent environments using immersive video. In *Proceedings of MODIE workshop at Mobile HCI'06* (Espoo, Finland, 12-15 September 2006).

[39] Strauss, A., and Corbin, J. *Basics of Qualitative Research: Techniques and Procedures for developing Grounded Theory*. Sage Publications Inc, 1998.

[40] Varoudis, T., Dalton, S., Alexiou, K., and Zamenopoulos, T. Ambient displays: influencing movement patterns. In *Proceedings of the 2011 annual conference extended abstracts on Human factors in computing systems* (New York, NY, USA, 2011), CHI EA '11, ACM, pp. 1225–1230.

[41] Weiser, M. Human-computer interaction. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1995, ch. The computer for the 21st century, pp. 933–940.

Appendix A

Questionnaires

System Usability Scale

	Strongly disagree				Strongly agree
1. I think that I would like to use this system frequently	1	2	3	4	5
2. I found the system unnecessarily complex	1	2	3	4	5
3. I thought the system was easy to use	1	2	3	4	5
4. I think that I would need the support of a technical person to be able to use this system	1	2	3	4	5
5. I found the various functions in this system were well integrated	1	2	3	4	5
6. I thought there was too much inconsistency in this system	1	2	3	4	5
7. I would imagine that most people would learn to use this system very quickly	1	2	3	4	5
8. I found the system very cumbersome to use	1	2	3	4	5
9. I felt very confident using the system	1	2	3	4	5
10. I needed to learn a lot of things before I could get going with this system	1	2	3	4	5

UbiComp questionnaire 18/10/2012

	Subject characterization								User Characterization
1	Age								
2	Gender								
3	Academic background								
		Strong disagree			**...**	**Strong agree**			
		-3	**-2**	**-1**	**0**	**1**	**2**	**3**	
5	Familiar with virtual environments								
	Experience								
6	I was able to interact with the system (application selection)								System
7	The environment was responsive to the actions that i initiated (or performed)								
8	I was confused by the logic of applications selection								
9	I was involved in the virtual environment experience								Immersion
10	I was so involved in the experimental task, that i lost track of time								
11	All my senses were completely engaged								
12	The visual aspects of the virtual environment involve me								
13	I was aware of events occurring arround me in the real world								
14	I was aware of my display and control devices								
15	I quickly adjust to the virtual environment experience								User Satisfaction
16	Moving around inside the virtual environment was compelling								
17	My interactions with the environment seemed natural								
18	My experience in the virtual environment seemed consistent with my real-world experiences								
19	At the end of the experience, i felt that moving and interacting with the virtual environment was proficient								

Appendix B

Plan of ongoing test study

Application Selection: Field and Second Life User studies

Study dimensions

Our study is designed on three dimensions:

- Users: we are going to observe their behavior, personal feelings and usage patterns.

- Environment: what are the similarity between the real and virtual environment

- Ubiquitous system (Second Life & Instant Places): We are going to assess the system responsiveness

Objectives or research questions

- What type of conflicts can appear in using this system?
- How do people mediate potential conflicting requests?
- User acceptance of the ubiquitous system logic
- Is there any similarity between the virtual and real environment?

- Is the system feedback (on displays and mobile phones) valued by users as a mean to foster social interaction?
- Proof of concept: does the system work for people? Do they understand the mixed initiative interaction model or scenario?
- Did the users feel immersed in virtual environment?
- How does using the prototype change people's behavior or allow them to do new things?

Usage situations

- At least 5 users. Users are within a coffee environment and after we explained them the system, they may start to issue the requests.
- We can also simulate requests.

System Logic

- Number of requests per application is stored.
- Applications are shown decreasingly, based on the number of requests per application
- Applications are shown with an interval of 1-2 minutes
- We can alternate the default behavior and show the apps with individual requests counts

Logic details

- The default behavior shows the scheduled apps
- After the last app is shown as part of the default behavior, another app will show the requests counts for each app
- After that, another app will present the requested apps in a decreasing order based on requests
- And, afterwards, according to the number of requested apps, the display returns to the default behavior.

Data sources

- Observation (writing down the social behavior of people interacting each other and with the display)
- Final interview about the system usage and how people succeeded to interact each other
- Questionnaires

Logs

(The data will show the stream of apps presented alternating between default behavior and user based approach)

- the index and app name of the default behavior
- the app requests
- the selected apps based on requests

Other notes

- We have 4 apps (football, posters, presence and activity stream)

Interviews

1. Did you encounter any difficulties in requesting your preferred app? (time, popularity, not understand how the system works)
2. Did you understand the role of feedbacks? Did they ease the understanding of the system functionality? (know the time to vote, know why the system chose that app to present)
3. Did you have any problems in understanding the mixed initiative approach: users vote and system decide? What are the drawbacks of it? What are its advantages?
4. Is this solution appropriate to increase the social interaction between participants? Or, it is a solution concentrated on individuals? (Is the system logic and approach that foster social interaction while interacting with the system?)
5. Did you need to talk to the other and establish an agreement? Or, you clearly understood how you can influence your turn, i.e., getting your preferred app on display?
6. Did users find any similarity between the virtual and real environment?
7. Did you feel immersed in virtual environment?
8. Can you describe your experience in the Second Life virtual environment?
9. How difficult was to understand the system and the way to influence how the apps are shown?
10. Can you give us a feedback on how to improve the logic of application selection?
11. What features would you like to have?

Table 1: Sequence of the experiment activities

Activity	**Description**	**Time**
Instant Places Tutorial	During this phase, the participants can watch the display without issuing any requests. In this session, they are instructed what is Instant Places and the role of each app. We will show them how to request and app by using a laptop or a mobile phone connected to the Internet. For instance, we show them how to use the presence app and activity stream. These two apps can show dynamic data. Subtasks: - Make a checkin an see your name in the Presence app and Activity stream app	10m
App Voting/ Requesting	They got an idea of each app and can already have some preferences. The vote session is started. Two sessions: - Do not tell the other what app you would like to see - You can share your opinions and agree which apps you would like to see	20m
Interviews	In the end, we ask participants about the usage of our system and the overall impressions	20m
Spare time	This time is for further explanations.	10m
Total:		**60m**