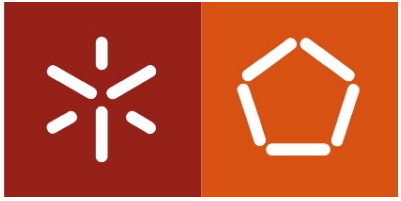
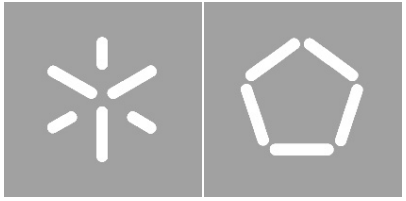


Universidade do Minho
Escola de Engenharia

Tiago Emanuel Oliveira Gomes

3D Virtual Environments' Generation

Outubro de 2013



Universidade do Minho
Escola de Engenharia

Tiago Emanuel Oliveira Gomes

3D Virtual Environments' Generation

Dissertação de Mestrado
Mestrado em Engenharia Informática

Trabalho realizado sob orientação de
Professor Doutor José Creissac Campos
Doutor José Luís Cardoso da Silva

Outubro de 2013

DECLARAÇÃO

Nome
Tiago Emanuel Oliveira Gomes

Endereço Electrónico
tg.gms89@gmail.com

Número do Cartão de Cidadão
13615192

Título da Dissertação
3D Virtual Environments' Generation

Orientador
Professor Doutor José Creissac Campos

Co-orientador
Doutor José Luís Cardoso da Silva

Ano de Conclusão
2013

Designação do Mestrado
Mestrado em Engenharia Informática

Universidade do Minho, / /
Assinatura:

Acknowledgements

I would like to dedicate this work to my family who were always present during my academic path and specially to my girlfriend Isabel Correia who was always ready to help and to give support.

I would like to give special thanks to my supervisors for all the advice, support, patience and availability throughout the dissertation. Particularly to José Creissac Campos for the expert guidance and to José Luís Silva for helping me to understand the APEX framework. Many thanks to Tiago Abade for his friendship and total availability during my passage through the University and specially during the course of this dissertation. Big thanks to Isabel Correia for recording the voices for the Asthma Game, and for the love, patience and encouragement during the dissertation. It would not be possible without her. Thanks to all of my friends for causing the good disposition required to carry out this work.

I would like to thank the Fundação para a Ciência e Tecnologia for their financial support.

Finally, I would like to thank my family for being my motivation. Thanks to my brothers for the comprehension and help. Thanks to my parents for the advice, for guiding me through my education, and also for the support needed to all of these years of study. And thanks one more time to Isabel Correia for being always present and for being my guiding star.

To all these people, I want to give my deepest gratitude. This work would not be completed, if it were not for them.

This work is funded by ERDF - European Regional Development Fund through the COMPETE Programme (operational programme for competitiveness) and by National Funds through the FCT - Fundação para a Ciência e a Tecnologia (Portuguese Foundation for Science and Technology) within project FCOMP-01-0124-FEDER-015095.

Abstract

3D Virtual Environments' Generation

The development and testing of ubiquitous environments (places enhanced with sensors, public displays and personal devices) usually presents high costs, both due to the need to acquire specific hardware (sensors, displays, etc.), and the need to use, or even to build, a space wherein the physical system will be implemented. Consider, for example, the impact of testing a new ambient intelligence system to provide information in a hospital or in an airport. It is hardly feasible trying to prototype the system in the target environment due to the costs (e.g. of redesign) and problems associated with such approach. The use of three-dimensional virtual environments then arises as a solution to this problem. Using them, it becomes possible to simulate the use of technology without needing to purchase hardware, and without interfering with the physical environments in which the final system will be installed.

Three-dimensional application servers such as SecondLife (secondlife.com) or OpenSimulator (opensimulator.org) provide an easy way to develop virtual worlds. A platform for the prototyping of ubiquitous environments is being developed at the Department of Informatics of the University of Minho, which is based on OpenSimulator: the APEX (rApid Prototyping for user EXperience) framework. At the moment, each new world has to be modelled manually, using an OpenSimulator compatible viewer, which makes this part of the process time-consuming and inefficient.

This project's objective is to study three-dimensional virtual environment modelling approaches, and to develop a module that integrates one of these approaches in the APEX framework to streamline the virtual worlds generation process.

Resumo

Geração de Ambientes Virtuais 3D

O desenvolvimento e teste de ambientes ubíquos (locais enriquecidos com sensores, ecrãs públicos e dispositivos pessoais) está normalmente associado a custos elevados, quer seja pela necessidade de adquirir *hardware* específico (sensores, ecrãs, etc.), ou mesmo pela necessidade de usar, ou até construir um espaço onde o sistema ubíquo será implementado. Considere, por exemplo, o impacto de testar um sistema inteligente de informação num hospital ou num aeroporto. É impraticável tentar prototipar o sistema no local destinado, devido aos custos (p.e. de redesenho) e dificuldades associadas. O uso de ambientes virtuais tridimensionais aparece como uma solução para este problema. Utilizando este tipo de mecanismos, torna-se possível simular a instalação da tecnologia sem que seja necessário adquirir o *hardware* e sem interferir com o espaço físico onde o sistema final será instalado.

Os servidores aplicacionais 3D como o *SecondLife* (secondlife.com) ou o *OpenSimulator* (opensimulator.org) proporcionam uma forma relativamente fácil de desenvolver mundos virtuais. Está a ser desenvolvida no Departamento de Informática da Universidade do Minho uma plataforma de prototipagem de ambientes ubíquos, chamada APEX (rApid Prototyping for user EXperience) que se baseia no servidor aplicacional *OpenSimulator*. De momento, cada novo ambiente virtual tem de ser modelado manualmente, usando um *viewer* compatível com o *OpenSimulator*, o que torna o processo demorado e pouco eficiente.

O objectivo deste projecto é estudar soluções para a modelação de ambientes virtuais tridimensionais, e desenvolver um módulo que integre uma dessas soluções na plataforma APEX, por forma a agilizar a criação de ambientes virtuais.

Contents

Acknowledgements **v**

Abstract **ix**

Resumo **xi**

Contents **xiii**

List of Figures **xvii**

List of Tables **xviii**

List of Abbreviations **xix**

1 Introduction **1**

1.1 Context 1

1.2 Objectives 2

1.3 Structure of document 4

2 Ubicomp Prototyping **5**

2.1 Introduction 5

2.2 APEX 6

2.3 OpenSimulator 7

2.3.1 Region Modules and OpenSimulator API 8

2.3.2 Environment generation problems 10

2.3.3 Conclusions 12

3 Developing Virtual Environments **14**
3.1 Introduction . . . 14
3.2 OpenSimulator Archives (OARs) . . . 14
3.3 Virtual environment modelling languages . . . 17
3.3.1 Scene graphs based languages . . . 18
3.3.2 Procedural modelling . . . 20
3.4 3D modelling tools . . . 22
3.4.1 Blender . . . 23
3.4.2 MeshLab . . . 24
3.4.3 Sweet Home 3D . . . 26
3.5 Discussion . . . 28
3.6 Conclusions . . . 30

4 Virtual environment development tool **32**
4.1 Introduction . . . 32
4.2 Using 3D Modelling tools . . . 34
4.2.1 Modelling the Building in SH3D . . . 35
4.2.2 SH3D and Blender . . . 36
4.2.3 SH3D and MeshLab . . . 38
4.3 Using OpenSim API . . . 39
4.3.1 The Region Module . . . 41
4.3.2 The User Interface . . . 44
4.3.3 Modelling the Building . . . 46
4.4 Conclusions . . . 47

5 Integrating the tool with APEX framework **50**
5.1 Introduction . . . 50
5.2 Architecture . . . 51
5.2.1 Remote Access . . . 51
5.2.2 Multi-user . . . 54
5.3 Summary . . . 56

6 Developing serious games with APEX framework **58**
6.1 Introduction . . . 58

6.2 Asthma 59
6.3 The Virtual Environment 60
6.4 The game 61
6.5 Evaluation 64
6.5.1 The user study 65
6.5.2 Results 66
6.6 Game redesign 68
6.7 Conclusions 70

7 Conclusions and future work **72**
7.1 Overall analysis 72
7.2 Results 75
7.3 Future Work 77

Appendices **79**
A.1 Questionnaire 79

Bibliography **84**

List of Figures

2.1 Logical architecture of the APEX framework (from [Sil12]) 6
2.2 Environment generation through a viewer (basic box) 11
2.3 Creating a wall with a window with 4 primitives 12

3.1 OpenSim Archives internal format 15
3.2 Scene graph example of a house 18
3.3 Procedural building phases 21
3.4 Manipulating 3D meshes with Blender 24
3.5 MeshLab in action [CCR08] 25
3.6 Sweet Home 3D interface 27

4.1 Building model blueprint 33
4.2 Using 3D modelling tools approach 34
4.3 SH3D Building development 35
4.4 Using Blender to transform the building model 36
4.5 Result for the solution SH3D+Blender 37
4.6 Using MeshLab to transform the building model 38
4.7 Result for the solution SH3D+MeshLab 39
4.8 Using OpenSim API approach 40
4.9 Triangle representation for angle calculus 43
4.10 OpenSim API approach interface 45
4.11 OpenSim API approach resulting model 47
4.12 Solution experience time results 48

5.1 Remote access architecture 52
5.2 Environment generator connection panel 55

5.3 Multi-user architecture 56
5.4 Complete solution sequence diagram 57

6.1 Mites in our homes 60
6.2 House of the asthma game 61
6.3 Pets in the bedroom 62
6.4 Question about dirty clothes 63
6.5 Survey results 66
6.6 Utility section results 67
6.7 Useful area bounded by barriers 69

List of Tables

3.1 3D tools relevant features 31

List of Abbreviations

2D	two-dimensional
3D	three-dimensional
API	Application Programming Interface
APEX	rApid Prototyping for user EXperience
COLLADA	COLLAborative Design Activity
CPN	Coloured Petri Nets
DAE	Digital asset exchange
DB	Data Base
DCC	Digital content creation
DLL	Dynamic Link Library
GNU GPL	GNU General Public License
GUI	Graphical User Interface
IP	Internet Protocol
LSL	Linden Scripting Language
L-System	Lindenmayer System
OAR	OpenSimulator region ARchive
OBJ	Wavefront OBJ
OpenGL	Open Graphics Library
SH3D	Sweet Home 3D

UML	Unified Modelling Language
VE	Virtual Environment
VR	Virtual Reality
VRML	Virtual Reality Modelling Language
WWW	World Wide Web
X3D	eXtensible 3D Graphics
XML	eXtensible Markup Language

Chapter 1

Introduction

1.1 Context

The rApid Prototyping for user EXperience (APEX) framework [SORF+10] was developed for prototyping ubiquitous computing environments. The framework uses a three-dimensional (3D) application server to provide a virtual environment were the envisaged ubicomp environment can be experienced, the OpenSimulator[1]. Typically OpenSimulator virtual environments are designed using the viewer tools also used to navigate the worlds. However, the experience of using the framework shows that the step of designing the virtual environment using this approach is a slow and laborious one.

When simulating real spaces enhanced with ubiquitous systems through virtual reality applications, such as SecondLife[2] or OpenSimulator, the creation and design of the environment itself is one of the most time consuming and demanding processes that developers have to face. This is because the contents in it have to meet high levels of quality (e.g. in terms of detail) in order to lead the user into a pleasant and realistic experience where real world objects can be easily recognised both by its appearance and behaviour. If, for example, the user cannot recognise a place in a virtual world where a ubiquitous system is being tested, results of the prototyping process will never be as reliable as they should be. Also, the

[1] `http://opensimulator.org/` (August 2013)
[2] `https://www.secondlife.com/` (June 2013)

most frequently used techniques for describing the virtual world are very low level (e.g. Open Graphics Library (OpenGL)) which means that the developer may have to gather large amounts of information in order to provide reasonable input. Thus, the process becomes slow and the virtual world requires great effort to be developed.

1.2 Objectives

The main objective of this project is to build a component for the APEX framework that makes it able to use a modern and agile technique to develop virtual reality environments. An alternative to achieve this objective is to load 3D models into the virtual environment, another alternative is to use the application server Application Programming Interface (API) to build the virtual objects. So, modelling techniques and existent markup languages that can be used for developing virtual worlds must be studied. Also, some existent tools that already use those techniques are going to be analysed. The chosen technique must be powerful and accurate enough to meet the needs of developing 3D virtual environments which can be used for implementing and testing ubiquitous computing environments. However, it must also be abstract enough to avoid the problems that are inherent to low-level languages, like having too many lines of code or having to program how the virtual scene is organised, and to streamline the process of designing the environment. Another important aspect is that APEX users need to build the environment incrementally, which means that it must be possible to add, remove or change objects in the virtual environment during its development.

An OpenSimulator server provides two main ways of loading virtual objects or 3D models into the environment: one of them is loading an OpenSimulator region ARchive (OAR) (see Section 3.2 for a detailed description of OARs) into the server, and the other one is loading a COLLAborative Design Activity (COLLADA) model into the server. COLLADA[3] is an XML based schema that can be used for exchanging 3D models between interactive 3D applications (see Section 3.3). Moreover, there is also the possibility of developing virtual objects through the OpenSim-

[3] `https://collada.org/` (July, 2013)

ulator API. This possibility can, in principle, be used to develop very efficient solutions, but it requires good knowledge about the application server API, which must be thoroughly studied.

Thus, there are three initial possibilities for solving the virtual environment generation problem in the APEX framework:

- Choose the most appropriate modelling tool and generate an OpenSimulator region ARchive (OAR). The input received from the modelling software must be structured in an adequate tool in order to generate the OAR. This solution can be very powerful, however the generation of the OAR can be quite complex due to the transformations that generating the archive might imply. Also, useful features inherent to the chosen language can be lost when converting it to generate the OAR file due to incompatibilities. This solution may improve the compatibility and sharing of the the generated 3D models between OpenSimulator versions, since they are based on an OpenSimulator native file format, however the generated models may not be as accurate as they have to, due to the conversions made.

- Generate a COLLADA model from the input received from the chosen modelling tool. This solution can ease the sharing of the virtual objects created because COLLADA is already a standard. However, COLLADA files have a quite complex syntax, and its components are not always compatible with other languages.

- Create a friendly interface to interact directly with the OpenSimulator Application Programming Interface (API). The OpenSimulator API is considerably large. It has several capacities that are not useful for a typical APEX user and would only introduce noise and make the tool more complex. The research must be focused in the relevant features. An interface like this, can be hard to implement, because it will imply developing a communication layer, since the API is local to the application server and the APEX framework is most often used in a remote context.

1.3 Structure of document

In this chapter, the problem and the objectives of this project were presented and briefly discussed. In the next chapters, the APEX framework and some virtual environment modelling techniques will be described as well as the conclusions and future work of this research. The dissertation document is structured as follows:

- Chapter 2 - Ubicomp Prototyping: describes the current state of the art, including the APEX framework and its virtual environment component which is composed of an OpenSimulator application server and a compatible viewer.
- Chapter 3 - Developing Virtual Environments: describes some alternatives to generate virtual environments in the APEX framework. OpenSimulator region ARchives (OARs), some important virtual environment modelling languages, as well as relevant 3D modelling tools, are described in this chapter.
- Chapter 4 - Virtual environment development tools: analyses the use of the tools and technologies presented in the previous chapter in order to provide solutions for the APEX framework virtual environment generation problem. At the end of the chapter the most suitable solution is chosen.
- Chapter 5 - Integrating the tool with APEX framework: in this chapter the integration of the previously chosen solution for the virtual environment generation problem with the APEX framework is described.
- Chapter 6 - Developing serious games with the APEX framework: describes the development and test of a serious game in the APEX framework. The costs of developing the virtual environment will be analysed too.
- Chapter 7 - Conclusions and future work: analyses the solutions found regarding the objectives defined for this research. Additionally, it presents some conclusions and proposals for future work.

Chapter 2

Ubicomp Prototyping

2.1 Introduction

Building ubiquitous environments has proven to be a tricky task, which involves problems, such as environment testing, redesigning or hardware costs. While building a ubiquitous environment, every decision must be well supported by previous studies, otherwise the project costs will grow very quickly. Moreover, decisions that have already been taken can be difficult to change, given that the costs involved in the redesign process can be very high. Also, the testing process of each new configuration can consume too much time.

Ideally we should be able to try every configuration before building the real system. In a virtual environment, real spaces can be pictured with acceptable precision. Furthermore changing decisions while developing the ubiquitous system involves much less effort than in the real world. 3D application servers play an important role in this issue. These tools provide a simple way to install and explore virtual environments with very low investment and with an acceptable quality. Thus design errors can be easily detected in the early stages of the project, and changed before deployment. In the next section, the APEX framework and its components will be presented. It uses a 3D application server to manage virtual worlds and its objective is to prototype ubicomp environments.

2.2 APEX

It was already said that ubiquitous environments' development often presents high costs and that prototyping such environments can introduce an effective way to avoid unnecessary efforts. This way, we can test those environments before their development and consequently, we can also prevent major errors or unnecessary costs. The APEX platform arises as a framework that supports the prototyping of ubiquitous environments [Sil12]. The framework goals are to help the rapid creation of virtual environments that mimic an environment and the ubiquitous computing technology within it. APEX's architecture consists of four main components (Figure 2.1):

- The virtual environment component that contains a 3D application server and a viewer in it. This component is responsible for the virtual environment. It allows the user to navigate trough the virtual world and sends information about user's actions and position. The application server chosen was OpenSimulator. It provides ways to create and manage the virtual environment. See Section 2.3 for more information about OpenSimulator.

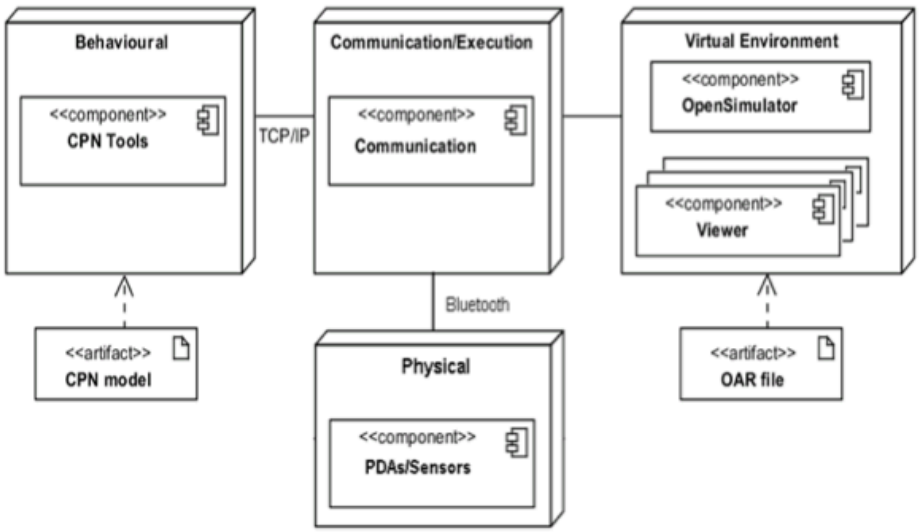


Figure 2.1: Logical architecture of the APEX framework (from [Sil12])

- The behavioural component is responsible for managing the behaviour of the prototype. It sends responses to the user actions based on a Coloured Petri Nets (CPN) model, where the prototype behaviour is described. This component uses CPN Tools[1] to support the management of behavioural models. The model keeps the modelled objects' state, so that the behavioural aspects of the objects can be kept independently from the virtual environment.

- The physical component is the one that manages connections with external devices. These devices can be smart phones, sensors or control pads for example. External devices can interact with APEX framework through this component, they can be used, for example, to move the avatar or to notify the user of the avatar state.

- The communication/execution component is a C# module that is responsible for loading the ubiquitous environment into the virtual world and for managing the exchange of information between the three other components cited before. Information about user actions is sent from the virtual environment to the behavioural models or physical devices through the communication component. Also, relevant information about behavioural models or physical devices state is sent to the virtual environment through this component.

2.3 OpenSimulator

OpenSimulator is a well known 3D application server. It can be used to create virtual environments to simulate scenarios of the real world. It is an open source project and is available for download on its website for the most frequently used platforms. It has the capability of being executed locally, so developers can easily customise it.

OpenSimulator can be accessed from a large set of client applications (viewers) that allow, as well, virtual environment manipulation through a Graphical User Interface (GUI). Viewers are the native tools for virtual environments creation on OpenSimulator. Viewers support the rendering task so that users can interact with the 3D application.

[1] `http://cpntools.org/` (June 2013)

An OpenSimulator virtual world is structured in regions. Each region can be accessed and managed independently. Not only 3D structures, but also sounds, can be loaded into a region. Moreover, an entire region can be loaded from a single file called an OAR. This constitutes an easy way of sharing regions. For more information about OARs see Section 3.2.

OpenSimulator provides an API to interact with external applications. This is how the APEX components interact with it.

2.3.1 Region Modules and OpenSimulator API

The other APEX framework components interact with the OpenSimulator server instance running in the virtual environment component through region modules. Modules are basically .NET Dynamic Link Librarys (DLLs) that are loaded when the OpenSimulator server is started. The OpenSimulator "bin" directory is scanned for DLLs on every initialisation in an attempt to find region modules stored there and load them to the current 3D application server instance.

Typically, region modules drive their execution by registering for events in the OpenSimulator instance (e.g. user logins, user movement, chat messages or even clock ticks). After catching an event the module executes whatever is needed to complete its task.

Currently, there are two different types of region modules: non-shared modules and shared modules. Shared modules are the most general and control the execution of all regions in an OpenSimulator instance. Non-shared modules are more specific and only control the execution for one of the regions in an OpenSimulator instance.

Region modules have a base interface that needs to be implemented for them to execute with no errors. A region module must implement INonSharedRegionModule or ISharedRegionModule for non-shared and shared modules respectively. Both of them extend IRegionModuleBase which is the base interface for any region module. The methods in this interface are listed and described below:

- Name - This method must return the name of the module.

  ```
  string Name { get; }
  ```

- ReplaceableInterface - This method provides stub functionality to region modules. This means that if no other region module implementing this interface is found then this method defines the default behaviour.

  ```
  Type ReplaceableInterface { get; }
  ```

- Initialise - This method is called right after the module finishes loading into the OpenSimulator instance. All configurations required for the region module to run correctly, may be set inside this method.

  ```
  void Initialise(IConfigSource source);
  ```

- AddRegion - This method is called when a region is added to the module. If the module is a shared one this method would be called for every region in the instance, but if the module is non-shared it will be called only once. The reference for a region scene must be set in this method so that it can be used later on the execution of the module.

  ```
  void AddRegion(Scene scene);
  ```

- RegionLoaded - This method is called when all the registered modules for a determined region finished loading. The difference between "AddRegion" method is that here we have all the functionality provided by the modules that were already loaded.

  ```
  void RegionLoaded(Scene scene);
  ```

- RemoveRegion - This method is called for any removed region. This can happen either by removing a specific region manually or in a module, or by shutting the server down where all the current regions are automatically removed.

  ```
  void RemoveRegion(Scene scene);
  ```

- Close - The close method is called whenever the OpenSimulator server instance is shut down.

```
void Close();
```

Moreover shared region modules have to implement one more method called "PostInItialise" that is invoked when all the regions for a server instance have finished loading. So it is called on each new region. It provides the opportunity to configure the module settings every time a new region is added to the current instance of the OpenSimulator server.

There are several other methods in the OpenSimulator API. It is possible to achieve almost everything that can be done with server commands or with an OpenSimulator compatible viewer. Here we are particularly concerned with the methods and classes used to generate 3D objects in the virtual environment. The main class for 3D object creation is called "SceneObjectGroup". This class represents a group of linked objects. It provides several methods for the object manipulation tasks such as scaling, rotating and positioning. Another important method is implemented in the "Scene" class and it is called "AddNewSceneObject". This method is used to make some instance of "SceneObjectGroup" visible in the respective scene of a region. Also we can choose whether to persist or not, in the OpenSimulator Data Base (DB), the object we are adding to the scene. If an object is persisted, then it will be always visible in the region until it is deleted even if the server is shut down or breaks. If the object is not persisted, then it will disappear every time the server goes down.

2.3.2 Environment generation problems

As it was said, 3D application servers like OpenSimulator can help to reduce the costs associated with the prototyping of ubiquitous environments. However, the virtual world development process associated with these tools is not as evolved as it could be. It is very slow and inefficient when compared to other virtual environment areas like the games' industry, which is constantly evolving and making use of the most powerful techniques. Rule based or graph oriented modelling are examples of such techniques. There are several tools using these techniques to build 3D models. Some of them are going to be covered in the next chapter.

The current tool used to develop virtual environments in the APEX framework is an OpenSimulator compatible viewer. The process of creating objects through

the viewer is quite abstract allowing the user to create almost any imaginable scenario. However this makes the tool little efficient when trying to develop specific environments.

The environment generation process using an OpenSimulator compatible viewer is made by creating simple primitives (prims). A prim is the smallest part of a structure in the virtual environment, and so, the most abstract too. It can be a box, cylinder, prism, sphere, torus, tube, ring, cone or a pyramid. Each prim can be mapped with a texture.

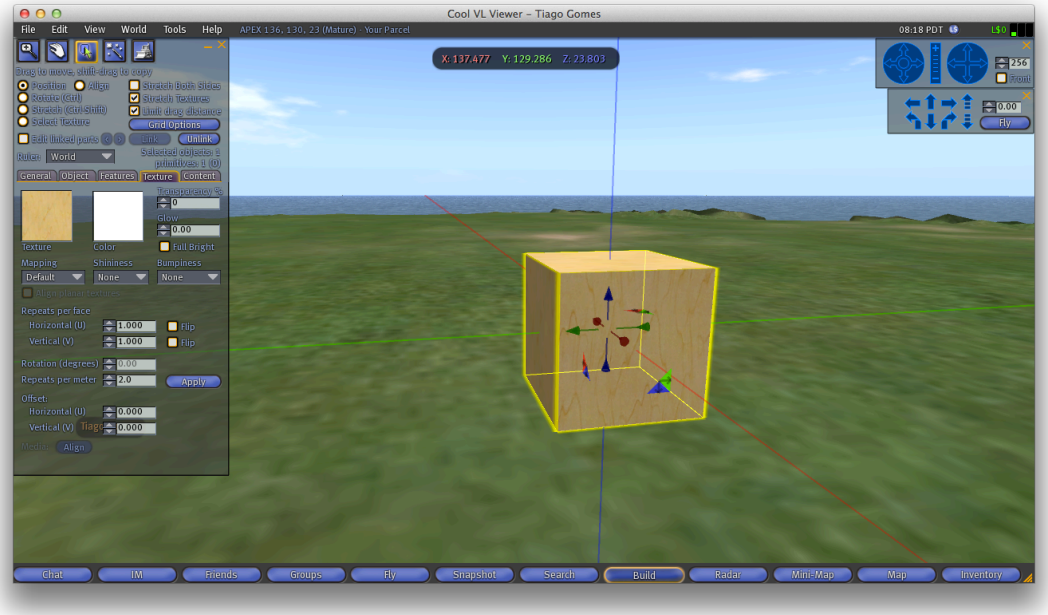


Figure 2.2: Environment generation through a viewer (basic box)

Creating a simple box with a cubic unit of OpenSimulator measures (Figure 2.2) is done by accessing the build option through the avatar menu and then choosing box as the prim type. The box dimensions can then be edited as well as a texture can be mapped. Changing the prim dimensions (X, Y or Z) is done by accessing the edit option of the prim menu to set the corresponding value in a text box or to drag and drop the corresponding dimension arrow. Mapping a new texture to a prim is also achieved through the edit option, clicking the texture button and choosing a texture. The objects are very manageable through this

creation process. However, to create a wall with a window, for example, it will not take less than 4 primitives for the most simplistic solution to be achieved (Figure 2.3). This makes the process very slow and inefficient when creating buildings.

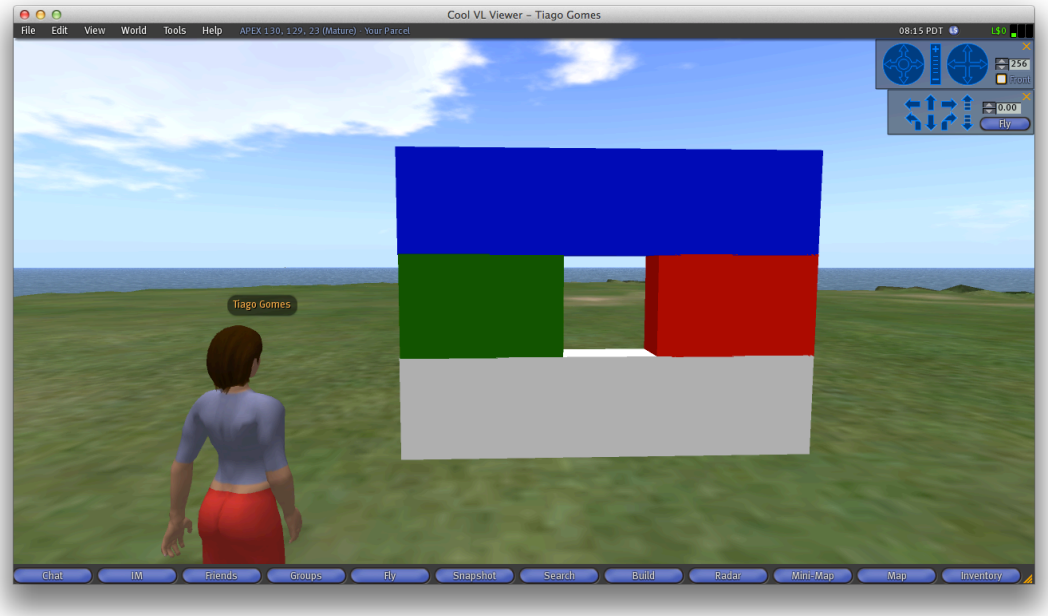


Figure 2.3: Creating a wall with a window with 4 primitives

The most recent viewers already have support for uploading COLLADA models. There are also 3D libraries on-line with several models (e.g. google 3D warehouse[2]). This enables the reuse of previously built models. Nevertheless, the building process, and in particular, the building of structures that can be populated with other objects, can be improved and benefit from the advantages of new modelling techniques and modern languages.

2.3.3 Conclusions

In this chapter the APEX framework and its virtual environment component, the OpenSimulator 3D application server were described. The OpenSimulator application server has an API that can be used to interact with the virtual environment

[2] `http://sketchup.google.com/3dwarehouse/` (September 2013)

programatically. This API can be used to develop solutions for the virtual environment generation problem since it provides methods to create objects in the virtual world.

Although APEX is a valuable framework for prototyping ubiquitous environments, the support for developing virtual environments is a feature of the framework that can be further improved, due to the issues related to the tool that is currently used to describe the environments, and to the fact that this tool is not tuned for the APEX needs. The most limited process is the development of the building's structure. The insertion of isolated objects can be satisfactorily achieved using models from on-line libraries. Relevant topics for virtual environment's development are described in the next chapter.

Chapter 3

Developing Virtual Environments

3.1 Introduction

In the previous chapter, the APEX framework and its components were described. The framework uses OpenSimulator to manage the virtual environments. It was said that the current tool for the virtual environment generation process does not satisfy the APEX framework needs. In this chapter some possible approaches to solve this problem are addressed.

3.2 OpenSimulator Archives (OARs)

OpenSimulator archives were created to ease the transport of whole regions from one server instance to another. The information in the OAR is not only about the shapes in the region, but also about the textures that cover them, the items contained in them, like sounds, and also about terrain (see [Jus09] for more details). All that information must be present in a single file because a whole region can be stored inside the OAR.

The .oar extension is used to identify the OARs, however these files are actually no more than common zipped tar archives (.tar.gz). By unpacking one of these files, one can see its internal structure. The folders and files contained in its root path are shown in Figure 3.1.

The content that can be found inside an OAR package includes the following

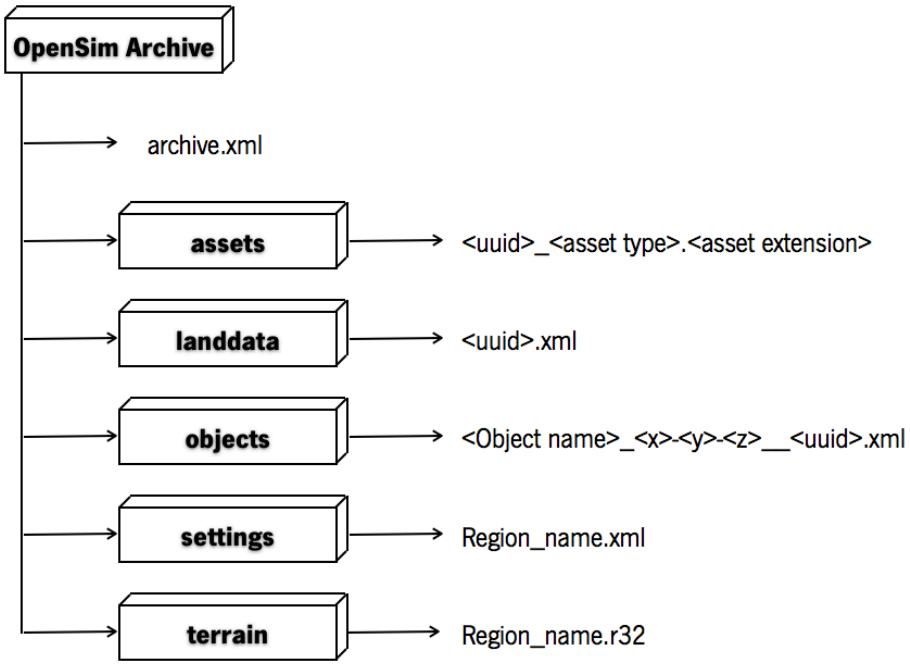


Figure 3.1: OpenSim Archives internal format

files and directories:

- The ”archive.xml” is known as the archive control file. This file contains information about version compatibility, a boolean value that specifies whether the OAR contains assets or not (see below), and a manifest of the regions included. It is written in eXtensible Markup Language (XML) format.
- The ”assets” directory contains all the assets in the archive. In the OpenSim context, assets are media files (e.g. textures, sounds, video and text) that are formatted and ready to be used in the virtual environment. There is no need to split them into regions because assets can be shared. Each asset name consists on an ”uuid”, which is used to identify the asset, and consequently must be unique, followed by the ”asset type”, which is used to identify the asset type, and finished with the asset extension which allows the asset to

be associated with specific editors.

- The ”landdata” directory contains all the parcels in the region. A parcel is an user created subdivision of a region. Information for each parcel is stored in separate XML files. The file name consists on as ”uuid” that unambiguously identifies the parcel.

- Each file in the ”objects” directory represents an object in the region. The file format used is OpenSim's XML2. Each file name is by default composed of an object name, followed by its position in 3D cartesian coordinates, and ending in its unique identification (uuid). Although, unlike asset file names, any component of its name can be changed without affecting the object itself. So these files can have any name because the information about the object is only taken from the XML.

- The ”settings” directory contains the settings information for the region in the XML format. These are the settings that can be accessed on the ”region settings” menu of any OpenSimulator viewer. For example, the ability to fly is a setting that can be found in these files. The file name will be the same as the region name.

- Finally the ”terrain” directory contains the terrain file for the region, stored in raw format. Its extension must be ”.r32” and the file name is also the same as the region name.

The OARs format was designed in order to overcome three main problems [Jus08]. First, to make it easy for people to read and write different OpenSim files within an archive. Since OARs are common compressed files, anyone can open them and manage their content. Moreover, the files within the archive are laid out to make it easy to perceive the different types of data. Second, to make it easy to compose two region archives into a single region archive. This means that the contents in one region can merge into another one just by copying the respective files from a directory to the other. Note however, that files in the terrain directory are an exception because one terrain relief would override the other. Hence this feature is more useful for archives that are collections of objects rather than whole

regions. Finally, to make it possible to compose archives from scratch. There is no obligation to create an OAR by saving the contents of an existing region. The user can create his own files outside of the system and compose the archive.

OAR archives are the most suitable solution to share virtual objects within OpenSimulator since they were created for it and are completely adapted to it. They store every single detail of a whole region so that no information is lost when sharing amongst different application server instances. However these files can be a problem in the context of the APEX environment generation due to the fact that every time they are loaded, the previous environment is overwritten, causing some information in other APEX components to be erased or unlinked. This issue would require the recreation of these information for the framework to work properly. Thus, consuming more time in the redesign process. One of the objectives of the APEX framework is to streamline the prototyping of ubiquitous environments.

Despite being possible to manually create OAR files, they are a technical file format and not the simplest way to build virtual environments.

3.3 Virtual environment modelling languages

An alternative to the archives discussed above (OAR) is the creation of virtual models from 3D modelling languages. Using modelling languages to create 3D models can provide a high level of customisation and accuracy. Although, the compatibility is imposed by the OpenSimulator ability to import the generated models, which currently is limited to COLLADA files. This problem can be solved by converting an input language format into the COLLADA format. Actually there are several tools that are able to make this conversion (e.g. Blender, MeshLab). Another advantage of the modelling languages is that the models can be simply added to the virtual environment without affecting any existent objects and there is no need to overwrite the whole region.

The most relevant modelling languages are going to be described below.

3.3.1 Scene graphs based languages

Scene graphs are a well known structure that is very useful when arranging the logical and spatial representation of a graphical scene. They can be used to represent 3D virtual worlds in computer graphics applications. A general definition of a scene graph is that it is a collection of nodes linked by edges and organised in a tree structure, so each node can have several children nodes but only one parent node. This is a hierarchical structure, which means that any operation applied to a parent is also applied to its children. This property is called state inheritance. The nodes in the scene graph store the information to manage the scene, while the edges link them in a way related to the spatial and semantic arrangement of the objects. A common feature of scene graphs is to group related objects into a compound object which can then be transformed as a single object.

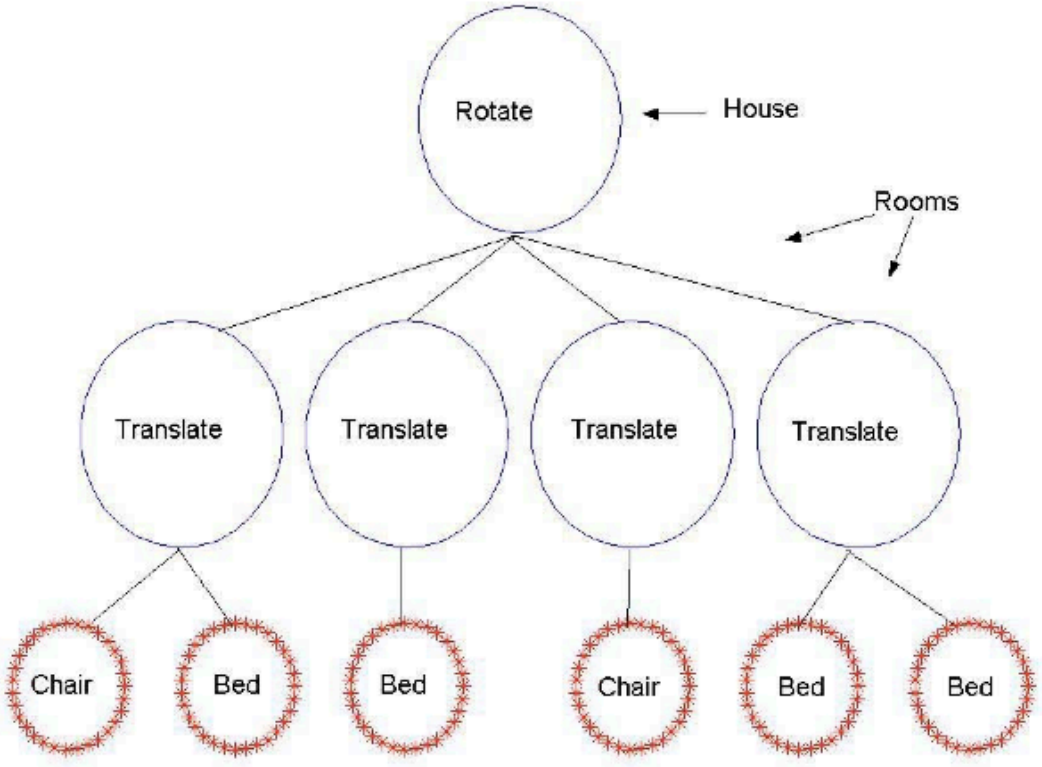


Figure 3.2: Scene graph example of a house

There are three different types of nodes in a scene graph:

- The root node is the first and it is the only one that does not have a parent node. All the other nodes are connected either directly or indirectly to the root node. This means that every leaf applies its operations, since they all derive from it.

- The internal or group nodes are the nodes that stand between the root and the leaf nodes. They are commonly used to preform a group of 3D operations like translations, rotations, scaling or shearing. Internal nodes describe the virtual world's state (position, orientation and size).

- The leaf nodes have no children nodes, so they cannot be parent nodes. They contain geometric data with the succession of operations of their ancestors (in a direct line from the root node) applied to them.

An example of a scene graph is shown in Figure 3.2. In this example the entire house is rotated in the root node, then every room is placed in their respective position by a translation operation and finally chairs and beds are rendered in each of the rooms.

Scene graphs are high level graphics languages that avoid procedural aspects of low-level ones like OpenGL [Woo03]. Scene graphs use a low-level graphics tool to render the scene, however they are completely independent from each other. The scene graph describes and updates the scene and indicates the content to be rendered by the underlying API.

Some languages that implement scene graph structures are described below.

Virtual Reality Modelling Language (VRML)

The Virtual Reality Modelling Language (VRML) is a standard for representing 3D models. It was created by the Web3D Consortium[1] to represent 3D interactive vector graphics on the World Wide Web (WWW). Despite using the scene graph's structure, it does not have a reference to a root node in its syntax. The root node is ambiguous and can be any of the other nodes.

VRML has default support for simple graphics primitives like cubes or spheres.

[1] `http://www.web3d.org` (April, 2013)

X3D

X3D is an XML based standard that is used for representing 3D computer graphics. It is the successor of VRML and features some extensions to its predecessor like the access to programming languages. It was also created by the Web3D Consortium. Due to the fact that it is based on XML, it benefits from some advantages like the freedom to define your own syntax. This feature makes the language very versatile and allows it to be easily adapted to any situation.

COLLADA

The COLLADA schema development was initiated by Sony Computer Entertainment[2] and the project's objective was to create a Digital asset exchange (DAE) format [AB06]. Nowadays many other Digital content creation (DCC) [EV01] tool companies are working together joining their expertise to improve the format. It was built using other open standards (e.g. XML, UTF8, XPath etc.) and it is also an open standard.

COLLADA defines an XML database what enables anyone to freely exchange COLLADA models without loosing any information. Its schema can be easily extended by end users for their own purposes. It was not designed to fit some specific needs but to be a media content holder for any target platform. It also supports all the basic features that a 3D interactive application needs, including shader effects and physics simulation.

The COLLADA format has a relevant advantage from the formats described before. It is the only format supported by OpenSimulator for now. So no conversions are needed in order to upload COLLADA files into OpenSimulator.

3.3.2 Procedural modelling

The generation of large 3D environments is a very time consuming task when developing virtual environment applications. In a large scene, there are sometimes very similar constitutes that have very similar building processes too. Usually, however, there are small changes between such constitutes. Consider for example

[2] `http://www.scei.co.jp/index_e.html` (April, 2013)

the development of a virtual environment representing a care home. There would be a great number of rooms in it, and every room would be slightly different from the other, but with same kind of objects and the same kind of sizes. An approach where every object is developed individually would be unfeasible, given the effort and time it would implicate, when we just want very similar rooms with some small differences. A possible approach for this problem is to provide a set of parameters to a function or algorithm that can generate those slight differences. This process is called Procedural modelling.

Figure 3.3: Procedural building phases

Using procedural methods for generating 3D environments has proven to be a good solution for problems like the one described before. Procedural modelling techniques require reduced human effort, by automatically generating 3D models [SC11]. However, the process needs guidelines (parameters) from the users to transform the objects as desired. The process is usually characterised by having a set of generation rules that transform groups of the most simple objects into most complex ones adding small changes to them, step by step. However, the results produced may have low graphical details quality which means that additional tuning might be required.

Fractals [Pen84], Lindenmayer Systems (L-Systems) [Tal96] or Generative modelling are some examples where procedural modelling techniques are used. For example, L-Systems are usually used in virtual reality applications for modelling plant ecosystems. Geospatial L-systems [CBSF07] are an extension to the standard L-Systems that incorporates geospatial awareness so they can be used in a geographic context like the development of virtual urban environments. An application that perceives the spacial relations between objects in an urban environment

can determine for example that a building wall cannot have windows because there is another wall too close.

3.4 3D modelling tools

3D modelling languages are the basis for the creation of virtual environments, although it is unthinkable describing a whole building by manually writing a 3D model using these languages. Nowadays, there are several tools for developing virtual environments which are based on these languages, however they do not imply writing the models manually, or even to have knowledge about the underlying language. Instead, they provide Graphical User Interfaces (GUIs) that receive the user operations and translate them into the underlying language.

3D modelling tools help people develop virtual models of 3D scenarios and support their edition and manipulation. A 3D model is composed by a polygon mesh or a set of points that give it its physical aspect. A user can add new points to the mesh, delete or position the existing ones, giving the model the desired shape. Additionally, textures can be mapped to the mesh points so that the model changes its appearance. 3D modelling tools usually have a list of features that support these operations and many other useful ones. Generated 3D models can be exported into files. There are several alternatives to express these models, as discussed in the previous section. Modelling tools may support some of these formats, either by importing or exporting, giving models compatibility with other 3D applications such as OpenSimulator.

There is a wide diversity of 3D modelling tools in the market. We can find both professional or more simple solutions for home use. Also, some are licensed under commercial licenses and others under open source licenses.

Here we aim to find a solution, composed by one or more tools, that can be integrated with the APEX framework in order to ease Virtual Environment (VE) creation. This solution should be a robust, scalable and easy to use tool for the development of buildings for prototyping ubiquitous environments. Thus, some criteria are going to be taken into consideration in order to serve the project needs. So, the solution must:

- be an agile tool for buildings development. The use of the APEX framework is based on the development of virtual buildings to test ubiquitous systems;
- have an intuitive and efficient interface granting ease of use. The use of the APEX framework shows that the current tool (OpenSimulator compatible viewer) is not the ideal solution in the buildings context. An easy to use tool will grant quicker results;
- be accurate enough to create scenarios with acceptable detail. The accuracy of the scenario is also a very important aspect in the APEX framework. Scenarios with low accuracy may lead the user to not recognising the place;
- support interoperability with OpenSimulator. This is a mandatory aspect, since OpenSimulator is the tool used to manage the virtual environment in the APEX framework;
- be composed by open source software so that we can easily get its source code and develop an integration layer with APEX framework.

This research is strongly based in these criteria. So, many other features of 3D modelling software, that are important in a wide range of cases, will not have a big emphasis in this analysis because they are not crucial for the project development and performance, and consequently we are not as concerned about them.

Some chosen tools are going to be described and analysed below, taking into acount the criteria cited above.

3.4.1 Blender

Blender[3] is an open source 3D computer graphics software project. It is licensed under the GNU General Public License (GNU GPL) so the public is free to use its source code. Blender is a multiple platform tool, it is available for download on its website for Windows, Mac OSX and Linux amongst other versions. Its initial stable release was made in 1995 and currently it is on its 2.68 stable version.

[3] `http://www.blender.org` (August, 2013)

Blender is a powerful tool that has many useful features for video games, visual effects and many other 3D interactive applications creation. Here we are particularly interested in 3D model creation and its strong compatibility with different 3D object file formats. It is fully compatible with the COLLADA format among several others. Some of the most relevant features for this project are:

- Texture management;
- Mesh manipulation;
- Support for importing and exporting several formats like COLLADA, Wavefront OBJ (OBJ), eXtensible 3D Graphics (X3D) or VRML amongst many others.

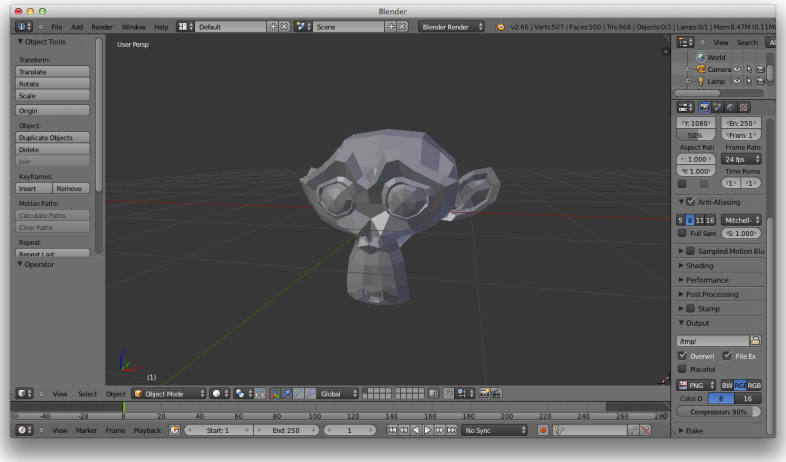


Figure 3.4: Manipulating 3D meshes with Blender

Blender has a large community of users and is widely accepted worldwide. Since Blender is an open source project we can easily improve it or create a plugin in order to integrate with the APEX framework, if needed.

3.4.2 MeshLab

MeshLab[4] is a system developed for graphical mesh processing. As with Blender, it is an open source project licensed under GNU GPL. It is easy to get its source

[4] `http://meshlab.sourceforge.net` (August, 2013)

code and develop a plugin to enhance MeshLab's list of features. MeshLab is highly based on the VCG Library[5] for all the core tasks on mesh processing. MeshLab is a cross platform product, it is even available for mobile platforms and is currently on its 1.3.2 version.

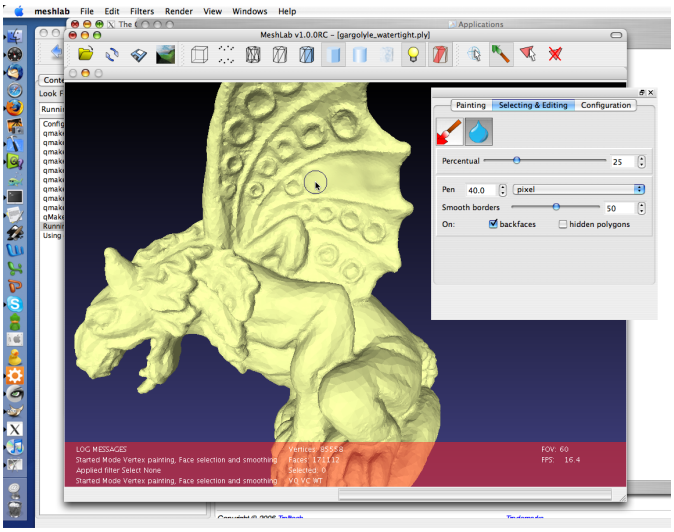


Figure 3.5: MeshLab in action [CCR08]

Amongst other important aspects, MeshLab was developed with three main objectives [CCR08] in mind:

- Ease of use - This means that no advanced 3D manipulation skills are needed to use MeshLab;
- 3D scanning oriented - Since there are already several strong players on mesh edition, MeshLab is particularly concerned about mesh processing tasks like cleaning, inspecting or converting meshes;
- Efficiency - MeshLab is a very efficient tool that can process large amounts of primitives on 3D scanning meshes.

[5] `http://vcg.sourceforge.net` (August, 2013)

MeshLab provides an extensive list of useful features on mesh processing [CCC+08]. Some of the most relevant characteristics of this project are listed below:

- Interactive selection and deletion features can be used to prune an imported mesh and adjust it to contain only what the user needs;
- The import and export features support many of the most used 3D file formats including COLLADA. Thus, any created or managed meshes can be transformed so that OpenSimulator can recognise them;
- Mesh clearing and remeshing filters can be used to remove the noise and redundancy in a model or to fill mesh holes in order to make it less complex and more lightweight;
- The measuring tool can be used to scale the imported mesh to the desired size by defining linear measures between pairs of points in the mesh.

This software is an efficient and useful tool for the mesh development. It can be used in the project to increase the objects' efficiency (remove redundant information) and also to give them compatibility with OpenSimulator since it supports a wide list of 3D object file formats.

3.4.3 Sweet Home 3D

Sweet Home 3D (SH3D)[6] was created for fast interior design. It is a fully tuned tool for building development. The program is aimed at people who want to quickly design their home's interior. Nevertheless, it can be used in numerous other contexts, due to its potential and ease of use.

Sweet Home 3D (SH3D) is also an open source project licensed under GNU GPL. Its source code can be easily downloaded at the website. The tool is very well supported and there are also some guides on how to develop plugins for SH3D. It is based on JAVA[7] and is a cross platform tool, it can be even used online on its website with no limitations. Currently the software is on its 4.1 version.

[6] `http://www.sweethome3d.com` (August, 2013)
[7] `http://www.java.com` (August, 2013)

Noteworthy is its ease of use with a low learning curve due to its approach for developing virtual scenes. The SH3D user interface is not a common 3D development tool interface, it is adapted to building development and this was an interesting aspect about this software. It is essentially based on drag and drop gestures and the most important is that a 3D scene is transformed in a two-dimensional (2D) plan where you can develop your building, thus dramatically reducing the complexity of the development task. Its interface is composed by four panels (see Figure 3.6):

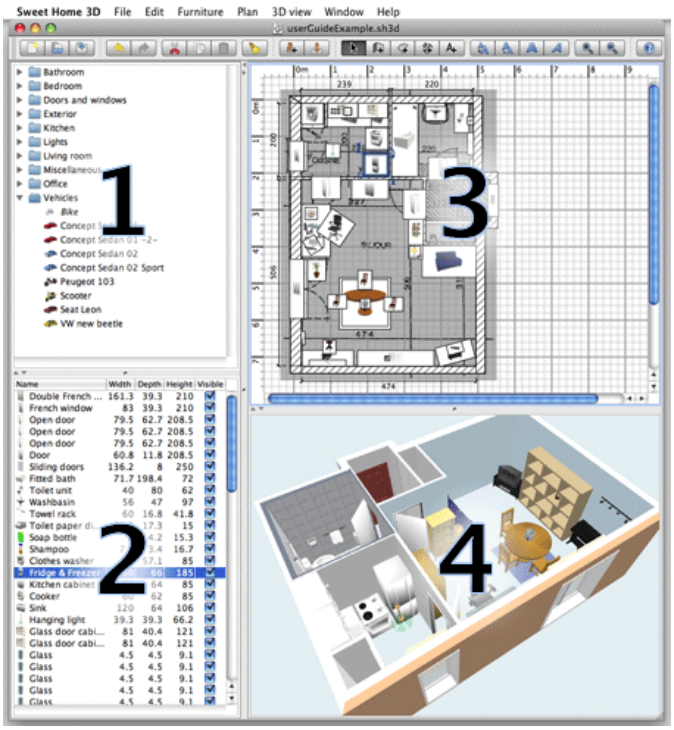


Figure 3.6: Sweet Home 3D interface

- Panel 1 is the furniture catalog. There, it is possible to find objects to add to the scene. A new object can be added to the scene by dragging it from this panel and dropping it on the home plan panel explained below. Moreover, it

is possible to add new objects to the furniture catalog simply by importing them.

- Panel 2 is the home furniture list. This list contains all the objects that were already added to scene. The objects can be selected and their characteristics edited.
- Panel 3 is the home plan. It is a 2D plan where the scene can be created. This is the main pane, every object on the scene must be added to it, either by dropping or creating it. It displays the building under construction as seen from the top.
- Finally, panel 4 is the home 3D view. This is the render pane, where the resulting 3D scene can be seen. It is updated in real time and the scene can be seen either from the top or from a virtual visitor's view at any chosen point.

Another nice feature of SH3D is that it can set a house blueprint as the home plan background. After setting the correct dimensions for the blueprint it is easy to pounce the walls, doors, windows or any kind of components in a building. This software is a mature player in the building development context, and so, it has a strong potential for the APEX virtual environment generation problem.

Despite having these useful features for virtual buildings development, Sweet Home 3D (SH3D) is not able to export generated models into COLLADA files. The 3D file format generated by SH3D is Wavefront OBJ (OBJ). So, an auxiliary software is needed in order to convert the output file from SH3D to an OpenSimulator readable format.

3.5 Discussion

Some possible approaches for the APEX framework environment generation were briefly studied and its main technical features were presented. Three main approaches were analysed:

- Use the application server API to build the necessary objects in the virtual environment;

- Use a modelling language to describe the virtual objects and generate an OAR archive to load them to the virtual environment;
- Use a modelling language to describe the virtual objects and convert them into a COLLADA file that is legible by the application server.

The use of OARs is going to be discarded. Despite the fact that it could be a very powerful solution and probably the most suitable for the 3D application server, OARs cannot be used to add objects to an existing region because the whole region is replaced when they are loaded. APEX users need to build the environment incrementally, which means that objects can be added, removed or changed in the virtual environment during its development. This would not be a problem if the environment component was the only one in APEX framework architecture, however it has also other components like the behaviour one that would loose information about all the current objects and would suffer from malfunction every time that a region was reloaded.

Still on the possible approaches, a consideration about the second one is that only scene graph based modelling languages will be taken into account from here on. This is because the procedural modelling approaches do not meet the needs of a standard user of the APEX framework. They were developed for large scenarios with low detailed objects, but when prototyping ubiquitous systems we need to provide very detailed scenarios to the end users in order to ease their immersion in the virtual world.

Thus, only two approaches are possible now, using 3D modelling tools or using the OpenSimulator API. Both of them are going to be properly analysed below.

Some 3D modelling tools have really interesting features for this project, however some other features in these softwares do not fit with APEX framework needs. For example, MeshLab has some nice tools for mesh optimisation, although it is unfeasible creating a whole 3D building using its user interface, moreover it would not add value over the current solution (an OpenSimulator compatible Viewer). Another example is that Sweet Home 3D has a very helpful interface and a well adapted approach for building creation, however it is not able to export generated 3D models into an OpenSimulator readable format. Blender provides an easy way to change the textures and do other manipulation tasks in a mesh and is able to

convert almost any 3D file format into another one, nevertheless its interface, as well as it happens in MeshLab case, leaves much to be desired in terms of buildings development.

The other alternative for solving the research problem is to use the OpenSimulator API. The OpenSimulator API provides methods to interact with the virtual environment. It is possible to make almost any operation that could be made through a server window or a compatible viewer. Adding a GUI to a region module developed using the API, can ease the use of the operations implemented inside this module.

3.6 Conclusions

Some powerful alternatives of 3D development applications were presented in this chapter, as well as the format of the OpenSimulator region ARchives (OARs), the standard files for storing and sharing whole OpenSimulator virtual regions. Moreover, a brief analysis of some virtual environment modelling languages that can be useful later in this project was made.

The 3D modelling tools described in this chapter have some limitations regarding the APEX framework virtual environment generation requirements. Despite having those limitations when they are applied in a standalone way, they can constitute some good solutions for the project's problem when applied together, joining their useful features. Filling the gaps of each other they can become a both efficient and well adapted tool which fills APEX framework needs, thus getting the best of both worlds. There are several more tools that can be used to develop virtual environments (e.g. Google SketchUp[8], Maya[9]), however some of them are not open source, or do not have support COLLADA files, or even do not add value to the tools described. The following table shows an evaluation of the tools described before taking into account the relevant features for this project. This evaluation was made considering the projects context and not a context of standard use of 3D modelling.

Looking at the Table 3.1 we can conclude that we need to join more than

[8] `http://www.sketchup.com/` (October 2013)
[9] `http://www.autodesk.com/maya` (October 2013)

Features	Environment Dev.	Ease of use	Accuracy	Collada support	Open Source
Blender			X	X	X
MeshLab		X	X	X	X
Sweet Home 3D	X	X			X

Table 3.1: 3D tools relevant features

one tool to get a complete solution that serves the project's needs. Thus, three hypothesis were considered:

- Joining Sweet Home 3D (SH3D) with MeshLab;
- Joining Sweet Home 3D (SH3D) with Blender;
- Joining the three tools.

Since all of the tools are open source softwares, we are able to get their source code and integrate with each other. These three potential solutions are going to be analysed in the next chapter. Moreover, the approach that uses the OpenSimulator API is also analysed.

Chapter 4

Virtual environment development tool

4.1 Introduction

In the end of the previous chapter it was concluded that there are two main possible approaches for the solution of the virtual environment generation problem in the APEX framework. One of the possible solutions is to use a group of 3D modelling tools that serve a set of requirements imposed by the APEX framework needs in order to make a joint coherent solution. An analysis of the relevant features of the modelling tools was made, and it was concluded that these tools could not be applied in a standalone way. Instead, they should be applied together in order to meet all the requirements.

The other approach is to use the OpenSimulator API. This approach assumes the creation of an OpenSimulator region module to interact with the virtual environment, and a user interface to call the respective methods on the region module. The user interface must be adapted to building creation so that it can streamline the virtual environment generation of a standard APEX framework project.

In this chapter, both of the approaches are going to be fully described and analysed. More than one solution may be presented for each approach. For analysis purposes, the model of a building will be used, so that the final product can reflect advantages and disadvantages for each solution. Moreover the time spent to build

the model for each solution is one of the most important aspects to the project, since the main objective is to streamline the virtual environment generation. The blueprint of the building can be seen in Figure 4.1.

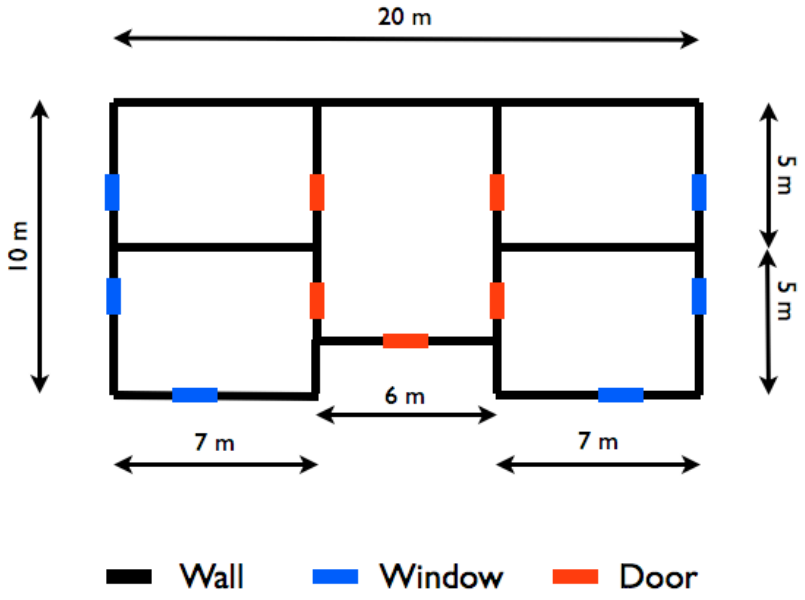


Figure 4.1: Building model blueprint

This is a simple medium scale building, with four rooms and a central hall. It has two windows in each of the front rooms and one window in each of the back rooms. Each room has an internal door that gives access from the central hall to the respective room. Moreover there is an entrance door that gives access to the building. The model of the building must be created using each solution so that results can be analysed and compared. At the end of the chapter, a comparison between the solutions will be made and the one that best fits the requirements will

be chosen.

4.2 Using 3D Modelling tools

The first approach that is going to be described is the use of the 3D modelling softwares to form a complete tool that meets the project requirements and can be adapted to the APEX framework. For this approach, three different solutions were found, although only two are going to be taken into account (see Figure 4.2). The one that joins the three tools was discarded because MeshLab and Blender play very similar roles in this task, thus using both in the same solution would introduce some unnecessary redundancy.

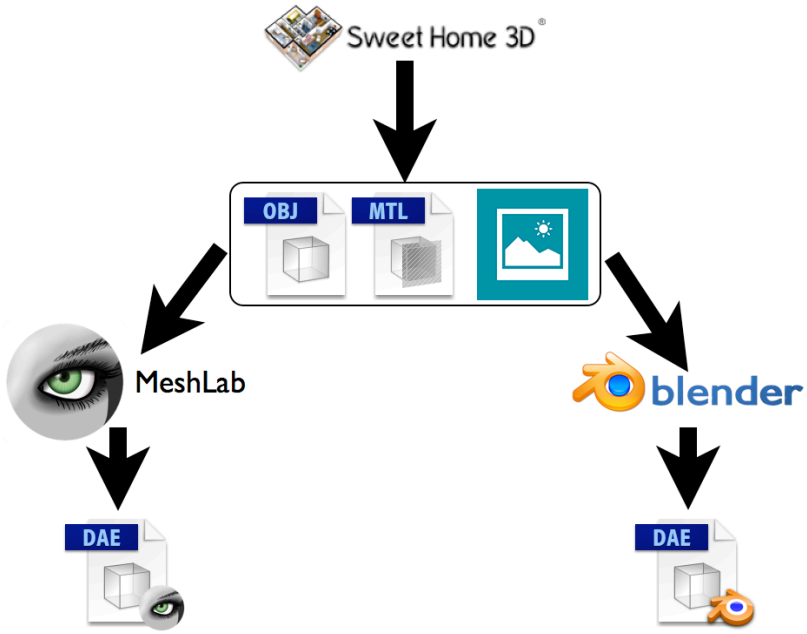


Figure 4.2: Using 3D modelling tools approach

All the solutions found have to include SH3D because this is the only tool that has a satisfactory approach to the development of buildings:

- Solution 1 - SH3D and Blender
- Solution 2 - SH3D and MeshLab

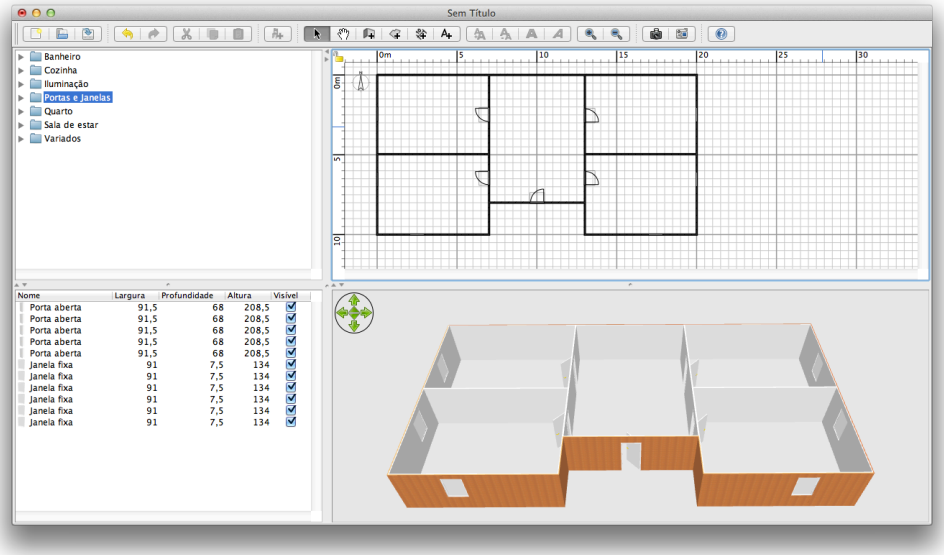


Figure 4.3: SH3D Building development

Each of these solutions is presented in the next sections. Both of them, however, start with a common task that is the definition of the building in SH3D.

4.2.1 Modelling the Building in SH3D

After launching the tool, the external walls started to be designed in the home plan taking the blueprint (Figure 4.1) measures into account. After that, the internal walls were designed and, to finish the building structure, all the windows and doors were added in the respective places. This was done by dragging and dropping them from the furniture catalog to the home plan panel. When the structure was complete, some textures were attached to the walls in order to give them a more realistic appearance. For this, each wall was selected in the home plan panel and its texture was edited by selecting the option ”change walls”. During this process it was possible to see the building evolution in the home 3D view panel in real time. This is a really useful feature since we can foresee the final aspect of the 3D model while we are designing the building. This enables us to quickly correct some imperfections that show up while the development is being made. The final aspect of SH3D when the building development was finished can

be seen in Figure 4.3.

In the end of the development process the 3D model was exported into the Wavefront OBJ (OBJ) format so that it could be used in the next by the 3D modelling software.

The process of developing and exporting the whole building model took about 7 minutes. This is going to be used as the base time for the solutions that are going to be described below using Sweet Home 3D (SH3D).

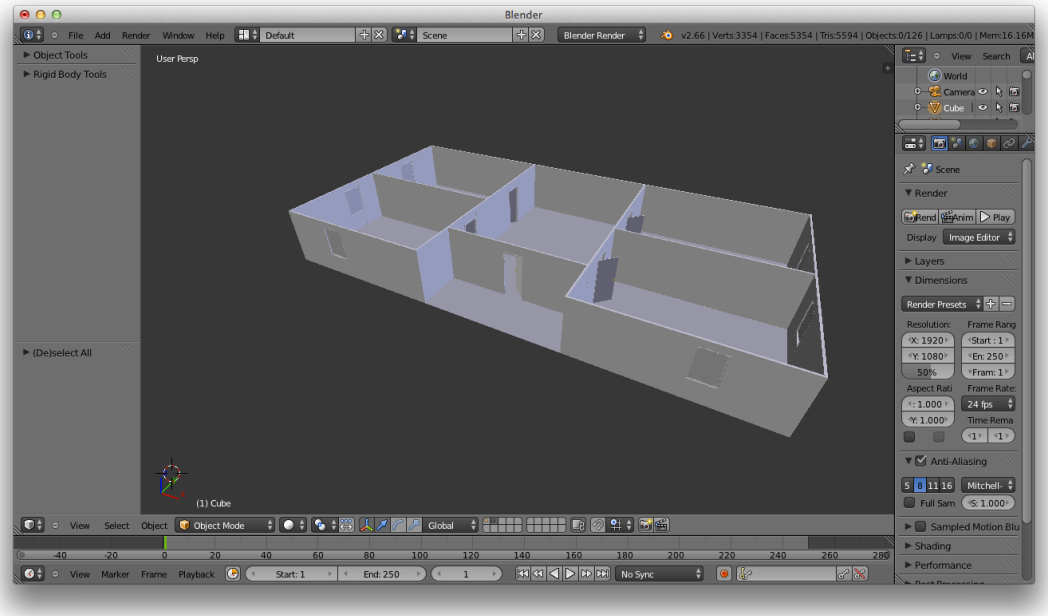


Figure 4.4: Using Blender to transform the building model

4.2.2 SH3D and Blender

The result of developing the building model in SH3D was a 3D model in the OBJ file format as it was said before. Some more files, such as texture images, were attached to the OBJ file. The next step in this process was to open the Blender application and import the resulting files into its environment. Here Blender plays basically a conversion role for the previously generated building model. After importing the 3D model, the building was rendered in the application main panel. By

selecting the export into COLLADA option, Blender created an output file containing the model of the building in the respective format. This task is illustrated in Figure 4.4.

With this step we obtained an OpenSimulator readable 3D model. So after that we just uploaded the resulting model to a local server instance using a compatible viewer. This process took about 6 minutes to complete, so the final time for this solution was about 13 minutes. Notice however that, if this solution is chosen, this time can be improved by automating the conversion process.

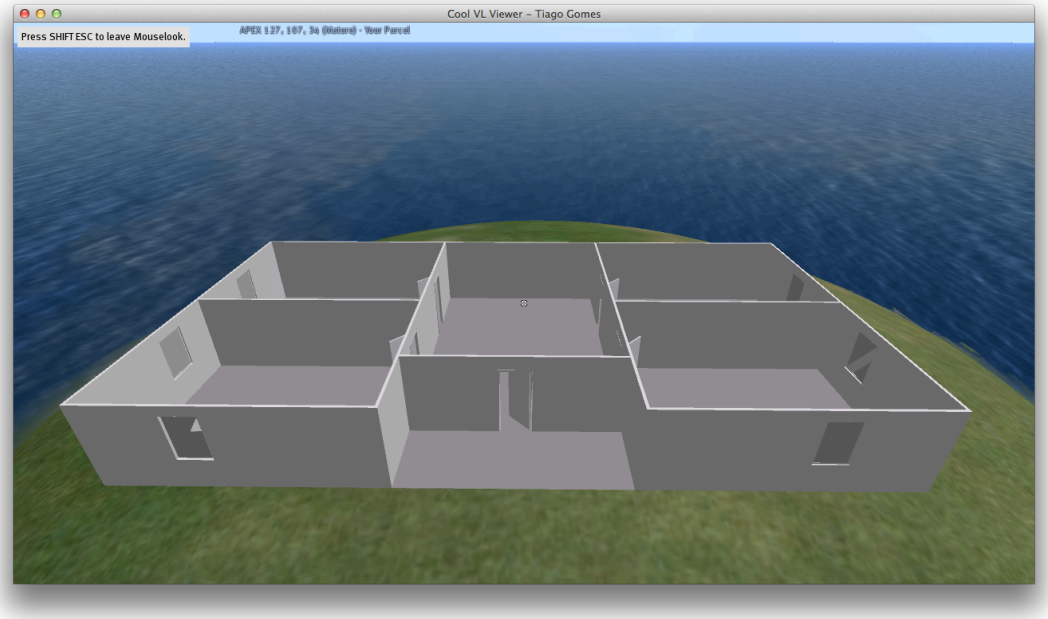


Figure 4.5: Result for the solution SH3D+Blender

Despite being an easy to implement solution, it revealed some problems in terms of compatibility of 3D file formats between Blender and OpenSimulator. The generated COLLADA model had a texture attached to the walls although it was not visible when uploaded to the OpenSimulator server. The resulting model had a homogeneous aspect and walls were hardly distinguished. Figure 4.5 shows the resulting model for this solution.

4.2.3 SH3D and MeshLab

This solution is slightly different from the one described before. The output (OBJ) 3D model from SH3D is also used as an initial model, although this time MeshLab is used instead of Blender for the conversion task.

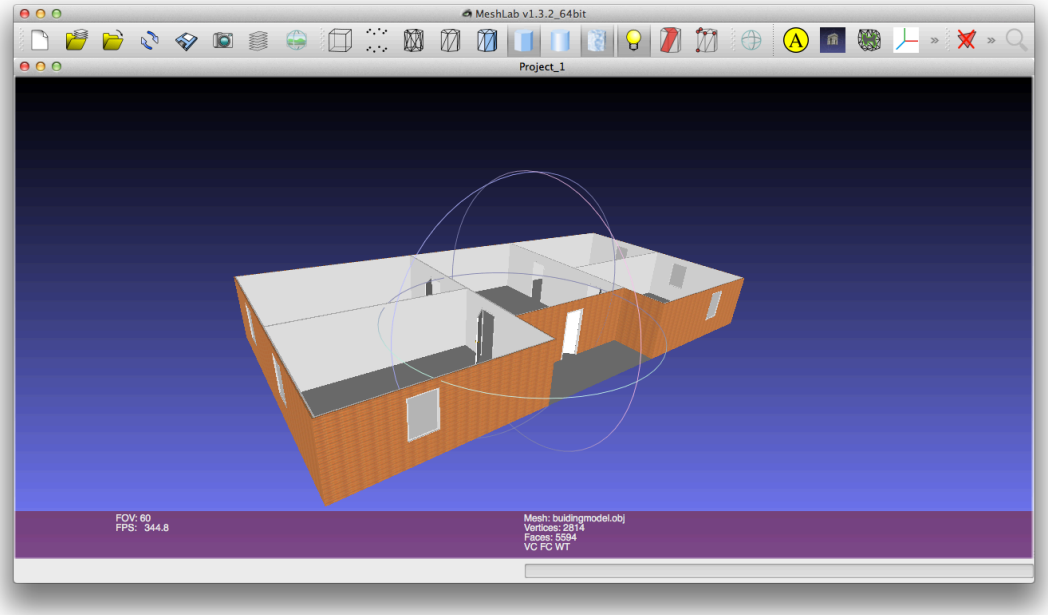


Figure 4.6: Using MeshLab to transform the building model

The process started opening MeshLab and importing the resulting mesh from the model of the building development in SH3D. As MeshLab also had support for OBJ file format meshes, the same file used in the previous solution was used here. After importing the model it was possible to see it in the MeshLab render panel (see Figure 4.6). After that, there were several mesh optimisation options available, but for the purpose we were just concerned about OpenSimulator compatibility. So the mesh was exported into a COLLADA file by choosing the option "Export Mesh as" and selecting the DAE extension. This process resulted in a COLLADA 3D model of the building that could be easily uploaded to an OpenSimulator server instance as it was made in the previous solution. Thus, after launching a local server instance the model was uploaded to it using a compatible viewer and it was possible to see the final result for this solution in action (see Figure 4.7).

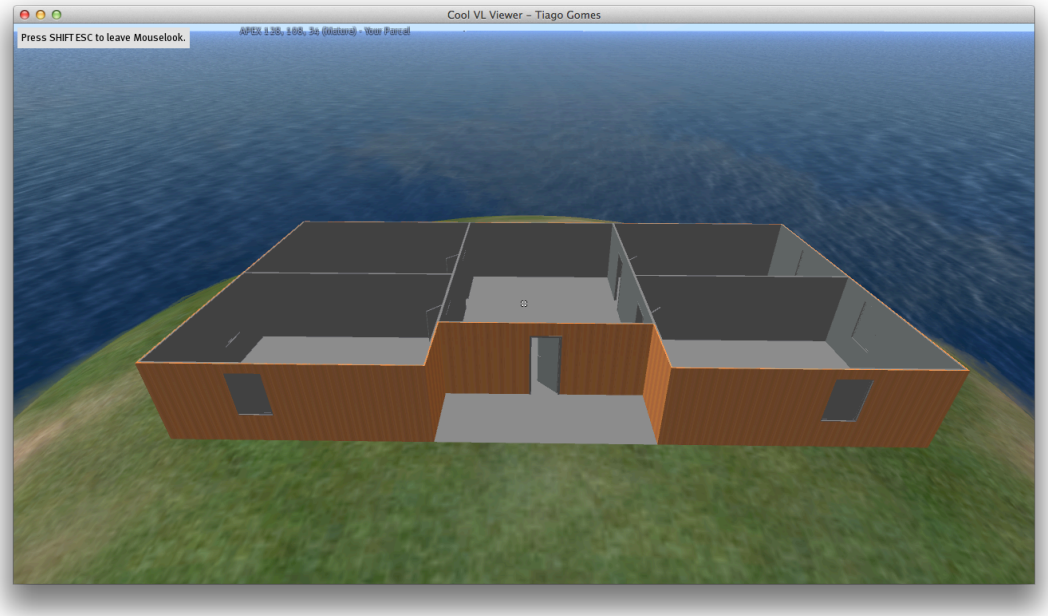


Figure 4.7: Result for the solution SH3D+MeshLab

The time elapsed since MeshLab was open until the result could be seen on OpenSimulator took about 5 minutes too, so the final time is very similar to the previous solution. Also, this time can be improved if this solution is chosen by doing the conversion task programatically.

This solution was also an easy to implement one. Although, the final result showed up some evident differences when compared to the previous one. Here the resulting COLLADA model seem to be fully compatible with OpenSimulator. The textures were uploaded correctly and the building had the same aspect as it was in SH3D. The result of this solution can be seen in Figure 4.7.

4.3 Using OpenSim API

The second approach described for solving the APEX framework environment generation problem is the use of the OpenSim Application Programming Interface (API). This approach can generate more customised solutions since it assumes the development of an OpenSimulator region module that interacts directly with

the virtual environment API. However, an user interface for describing the virtual objects (e.g. buildings) has to be developed in order to streamline the process and make it more user friendly. It is unfeasible trying to describe a 3D building model just by calling OpenSimulator API methods in a command line for example. Figure 4.8 shows the approach architecture.

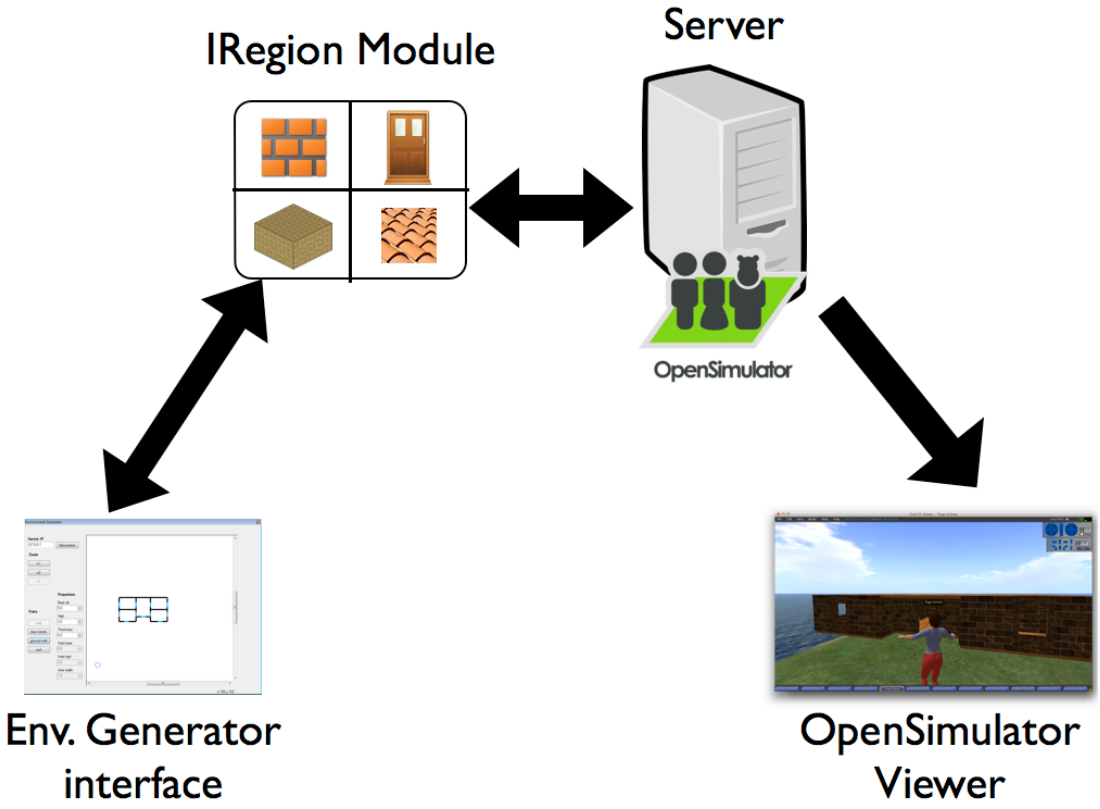


Figure 4.8: Using OpenSim API approach

The building process for this approach was structured in four basic tasks. Each one of them related to the creation of a common part of a building:

- Walls - The wall is probably the most common part of a building. For the approach's purposes a wall is going to be defined by two 2D endpoints, a height, a bottom altitude and a thickness value. The two endpoints define the 2D start and end coordinates for the wall, respectively. The height value defines the wall's height from bottom to top, the bottom altitude value

defines the altitude where the bottom of the wall is aligned, and the thickness value defines the thickness of the wall, as the names suggest.

- Holes - Holes are used for describing both windows and doors. Both of these parts are composed by a wall with a rectangular hole in its middle. So the parameters that define a hole part are the same that define a wall plus some more that define the hole dimensions and vertical positioning. The added parameters are the hole bottom that defines the distance between the wall bottom and the bottom of the hole (e.g. this parameter should be 0 for a door), and the hole width and height that define precisely the width and height of the hole inside the wall.

- Planes - Planes are simple rectangular structures that define grounds or ceilings in a building. Despite being represented by a rectangle they are also defined by two points in the 2D view. The first one is the top left corner point of the rectangle and the second one is the bottom right corner point. There are two more parameters needed to define a plane. The bottom altitude that represents the base altitude of a plane and the thickness that represents the plane thickness. Here the thickness is defined in the altitude (Z) axis.

- Roofs - In the virtual environment, roofs are represented as pyramids. Roofs are also defined as a rectangular structure, just as planes. Although, there is one more parameter needed to define a roof when comparing to the planes: the roof height, that is, the distance from the bottom altitude to the peak of the pyramid. Also there is another parameter that is not needed when comparing to planes, the thickness.

All the parts cited before must be sized in the OpenSimulator dimensions. This means that a measuring unit corresponds to an OpenSimulator unit of measurement too. The two main components for this approach are described in detail below.

4.3.1 The Region Module

As it was said before, this approach for solving the APEX framework problem requires the creation of an OpenSimulator region module. Since the building pro-

cess is divided into four basic components of an edifice, the development of the module started with the implementation of each one of this components. A class was created for each component containing all of its parameters and a method for rendering the corresponding object in the virtual environment. The region module created implements the INonSharedRegionModule interface, so the interface's methods had also to be implemented. Here the non-shared interface was chosen because we want all the environment generation functionality attached to a single region and not to a group of them.

An important aspect on the interface methods was the "Initialise" method implementation. The need to maintain a reference to the objects that were already deployed to the virtual environment appeared soon. Some features like removing existent parts have to refer to an already created object, so its reference is needed. To solve this problem a list of the current objects was created. This list is initialised in the "Initialise" method and is updated every time an object is created or deleted.

All of the four parts that compose the building process are defined by two points and a couple of other parameters. However, OpenSimulator has a different approach for object representation. In OpenSimulator, every object is represented by one centre point (object position), three more parameters corresponding to the sizes in the three axis (X, Y, Z) of the 3D space (object scale) and an angle (object rotation). The process of converting the object's parameters into OpenSimulator compatible parameters is achieved through trigonometric transformations. Which transformations are used depends on the object that is being created. For example, to transform a wall it is necessary to:

1. Calculate the distance between the two points that define the wall so that it can be mapped to the value of the X axis in OpenSimulator. The formula used to calculate the distance is derived from Pythagoras theorem:

$$d = \sqrt{\Delta x^2 + \Delta y^2}$$

   The values of Δ are calculated by the difference between the the respective coordinate from the start point to the end point. For example Δx is equiv-

alent to $(x_2 - x_1)$, x_1 being the X coordinate for the start point and x_2 the X coordinate for the end point.

2. Calculate the middle 3D point (x,y,z) between the two points that define the wall so that it can be mapped to the object centre point on OpenSimulator. The formula used to calculate the middle point is as follows.

$$P = \left(\frac{x_1 + x_2}{2}, \frac{y_1 + y_2}{2}, b + \frac{h}{2} \right)$$

   In the previous formula b represent the bottom altitude of the wall and h is the height of the wall.

3. Calculate the angle formed between the X axis and the line segment between the two endpoints of the wall so that we can get the rotation of the wall. The triangle internal angle formula can be used for this (see Figure 4.9).

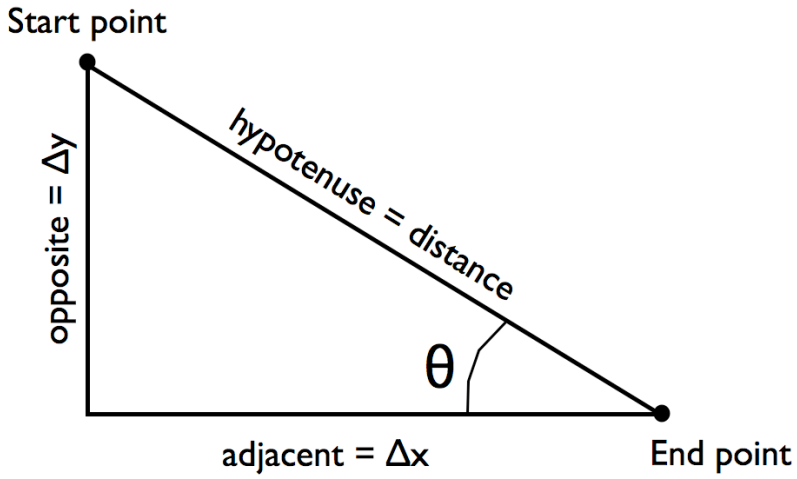


Figure 4.9: Triangle representation for angle calculus

$$\theta = \sin^{-1}\left(\frac{\Delta y}{d}\right)$$

Thus, this formula returns the rotation of the OpenSimulator object that has to be applied around the Z axis.

4. Finally, when the object position and rotation are already calculated, it is only needed to finish the object scale calculus. The X value of the scale is already calculated by the distance, only the Y and Z values are missing that are directly mapped by the wall thickness and height respectively.

As it was said, the transformation process depends on each component. Similar tasks have to be made to transform the other components into OpenSimulator readable ones.

4.3.2 The User Interface

The region module created is a key piece on the current approach, although, it would be useless without a user interface that can efficiently transform simple user actions into the required elements on the virtual environment. It was already said that Sweet Home 3D (SH3D) has a very efficient approach due to its user interface. It simplifies 3D building development by decreasing the number of dimensions to 2 in its home plan panel. Thus, the interface for this approach was developed taking into account that fact, so that it can also provide an easy to use and intuitive interface to the users. Although the user interface that was created does not provide a 3D view of the objects as done in SH3D, the virtual objects created are added to the OpenSimulator virtual environment in real time. Hence, it is possible to see the results just by connecting a compatible viewer to the server instance.

The user interface created can be seen in Figure 4.10. It is composed of four distinct parts:

1. Zoom - The zoom tool allows the user to zoom in or out over three levels. It has a button for each level of zoom. The first level resizes the design panel

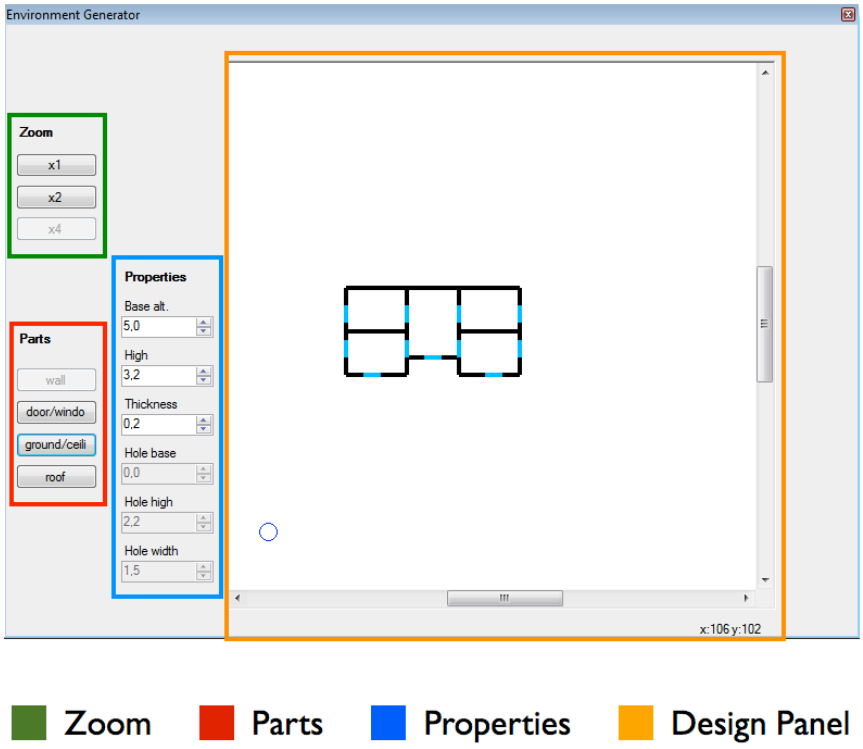


Figure 4.10: OpenSim API approach interface

area to the total area of a whole region, while the second and third levels resize the panel area to a half and a quarter of the region size respectively. Thus, the user can design more precisely some parts of the building by zooming in and can also design quickly some parts that require less precision.

2. Parts - The parts selector allows the user to select the type of part he wants to add. As it was said before the building process is composed of four parts, so there are four buttons in the parts selector, one for each part. When the user selects a part the properties panel adapts the parameters to fit the part characteristics.

3. Properties - The properties panel shows the current part parameters and their values while the user is designing. Each part has different characteristics so the properties panel is refreshed on each part selection. The properties

panel also lets the user customise the objects that are being designed. It is possible to change each parameter's value before designing an object.

4. Design panel - The design panel is the place where the user develops the buildings. It provides a 2D representation of the objects. Each part of a building has a different representation so that the user can distinguish them. Walls are represented by black lines, holes are represented by blue lines, plans are represented by grey rectangles and roofs are represented by orange rectangles. Moreover, the design panel provides the ability to delete already designed objects by right clicking on the mouse over the part that has to be removed and choose the option "delete".

   To improve accuracy when designing in the panel two features were implemented: position preview and position tracker. The position preview feature is achieved by a small blue circle that follows the cursor and converts its position into the resulting location in case of left clicking the mouse. The position tracker is a label located on the bottom of the panel that shows the current position of the cursor.

4.3.3 Modelling the Building

After developing the region module and the interface for this approach, a test was made with the building blueprint in Figure 4.1. The region module created was placed in the binaries folder of the OpenSimulator server package. The experience started launching the OpenSimulator server. The server instance loaded all the modules placed on the binaries folder including the module created for the environment generation. When it finished loading, the interface showed and it was ready to start designing the building blueprint. But before starting the design process, an OpenSimulator compatible viewer was launched and logged into the local server so that we could see the virtual environment.

The design process was very intuitive. The blueprint dimensions were easily treated due to the interface positioning features and all the parts were designed without difficulty. Only two mouse clicks for each part are required if using the default parameters for the building parts. Moreover, it was possible to check every

step of the building development through the OpenSimulator viewer in real time.

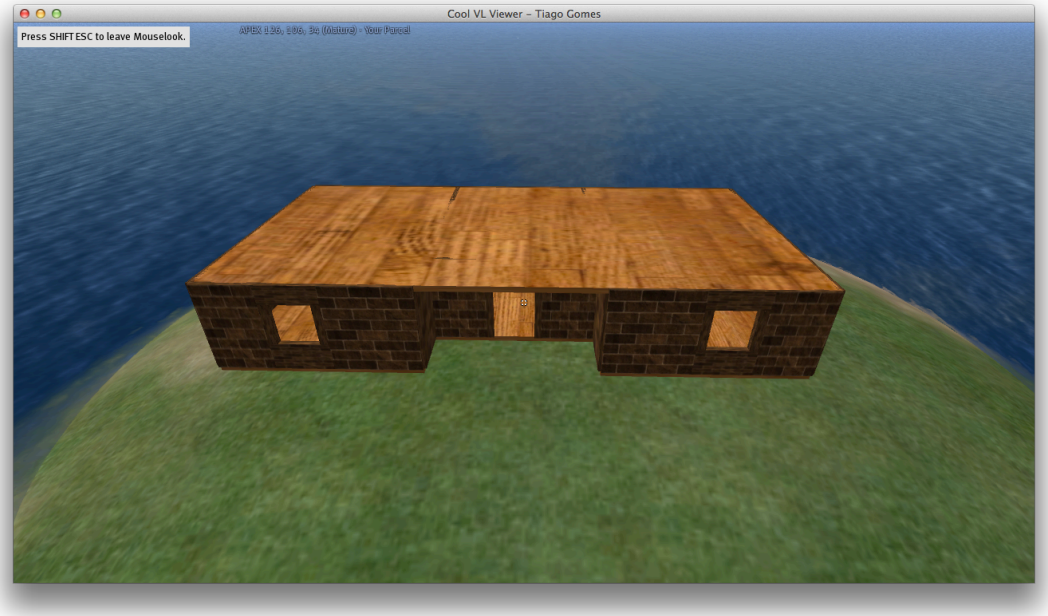


Figure 4.11: OpenSim API approach resulting model

The whole process took about 6 minutes to complete and the resulting model was quite accurate and nice looking (see Figure 4.11).

This was the solution that took more time to develop. Although the results of the experience in terms of time showed that the time elapsed is significantly better. Moreover this solution showed a high compatibility level and no issues were found on the resulting model of the building.

4.4 Conclusions

In this chapter three solutions for solving the environment generation problem in APEX framework were described and tested. Each one with specific peculiarities. Some of the solutions presented required more developing time and some other required more knowledge about 3D model manipulation. All of them met the APEX framework requirements. There were different execution times when experiencing each solution on the creation of the building represented on the blueprint in Figure

4.1. The time elapsed to create the 3D objects on the virtual environment is a very important aspect on the APEX framework context, since the main objective of this project is to streamline the virtual environment generation. The execution times are shown in Figure 4.12. These times are indicative and further study would be necessary to have reliable results. In any case, the choice is not only based in the numbers, but also in the integration aspect provided by the implemented solution.

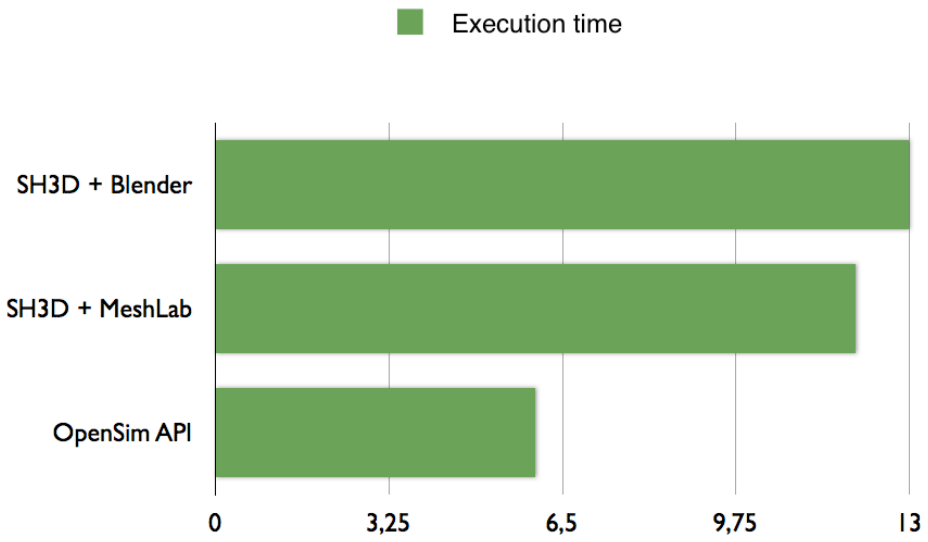


Figure 4.12: Solution experience time results

Looking at the results we can see that the two solutions generated from the first approach have very similar times, rounding twelve minutes. Both of them were quickly implemented since they are based on the use of external tools. However the solution with Blender showed some compatibility issues in the textures of the generated model. The solution derived from the OpenSimulator API approach had a very low time when comparing with the other solutions. This is mainly due to its approach. The solution tries to take the best from SH3D interface, that

is a mature software for the buildings development, and mix it with the APEX framework needs to reproduce its interface. Also, the fact that the objects being developed are created in the virtual environment in real time is an aspect that confers high agility to the solution, because the user is always seeing the result of his actions.

The SH3D interface adds value to the solutions generated from the approach with 3D modelling tools. Although, SH3D has also some aspects that are not so good in the APEX framework context. For example, the doors created on SH3D are static objects in the model and cannot have programatic movement as it can be done with OpenSimulator scripting. Conversely, the approach with the OpenSimulator API can be improved to attach a default script on each created door to simulate the door opening and closing movement dynamically.

The fact that two of the presented solutions are based on external tools, can be considered a negative aspect because these solutions are dependent on the development of the tools that are involved in its making. Moreover, the OpenSimulator API approach can offer high customisation, since the API provides a way to implement almost all of the actions that can be done using an OpenSimulator compatible viewer.

Hence, according to the reasons presented, the choice of the solution for the APEX framework environment generation problem lies on the solution using the OpenSimulator API. The current solution was not fully adapted to the APEX framework context yet. It was tested in a local environment and had not already a way of connecting to a remote OpenSimulator server, for example. It was already said that the remote connection is the standard use of the framework. So, some modifications and improvements had to be made in order to overcome these issues. The process of integrating the tool with the APEX framework is fully documented in the next chapter.

Chapter 5

Integrating the tool with APEX framework

5.1 Introduction

In the previous chapter some solutions for the APEX framework virtual environment generation problem were presented and tested. At the end of the chapter, the three solutions were analysed and the most suitable one was chosen. The choice lied on the solution that uses the OpenSimulator API. This solution showed its potential while it was tested with the house blueprint in the previous chapter. However, all the tests were made in a local context. Every tool instance was placed in the same machine, from the OpenSimulator server to the environment generator interface. It was already said that the APEX framework is frequently used in a remote way. So, in order to integrate the solution with the APEX framework, a network layer has to be inserted between the user interface and the other components. Moreover, the tests made to the chosen solution did not comply with a multiple user context. Some virtual environments can be very large and also composed of several buildings. A single developer approach to build such an environment should not be imposed. To make it possible to do cooperative work in the APEX framework, the environment generator must provide mechanisms to allow multi-user development. Thus multiple clients should be able to connect to a single region through the environment generator. This is another integration

feature for the chosen solution.

In this chapter, the features developed for the environment generator's integration with the APEX framework are going to be described and justified.

5.2 Architecture

As it was said before, the process of integrating the environment generation tool with the APEX framework assumes the development of some new features. These features are mainly required because the environment generation tool has to fit the APEX context. Some changes had to be made in the tool in order to achieve those features. Basically, the changes made are divided into two groups:

- Remote access - This group of changes is related to the feature of accessing the APEX framework remotely. That is, logging in into the OpenSimulator server and making use of the environment generator through a network connection, not needing to be at the same machine as the server.
- Multi-user - This group of changes is related to the support for multiple access and simultaneous use of the environment generator. In other words, the tool must allow cooperative work. APEX users should be able to connect to the environment generator simultaneously and develop the virtual environment together.

These groups are directly associated with the desired features for the tool to integrate with the APEX framework. Both groups are going to be fully described in the next sections.

5.2.1 Remote Access

To provide remote access to APEX framework users, some changes were made to the basic architecture presented in Figure 4.8. The environment generator interface has now to be separated from the server by a network layer (Figure 5.1). This means that the methods for the building parts creation cannot be called directly, but must be called through a network connection. So, a communication

dialect was developed. It was already said that the building parts are divided into four different structures. Thus, seven different kinds of messages were created in order to access remotely to the module features. Two of them for the connection establishment and closing, one for the objects deletion and the other four were created for each of the different building parts.

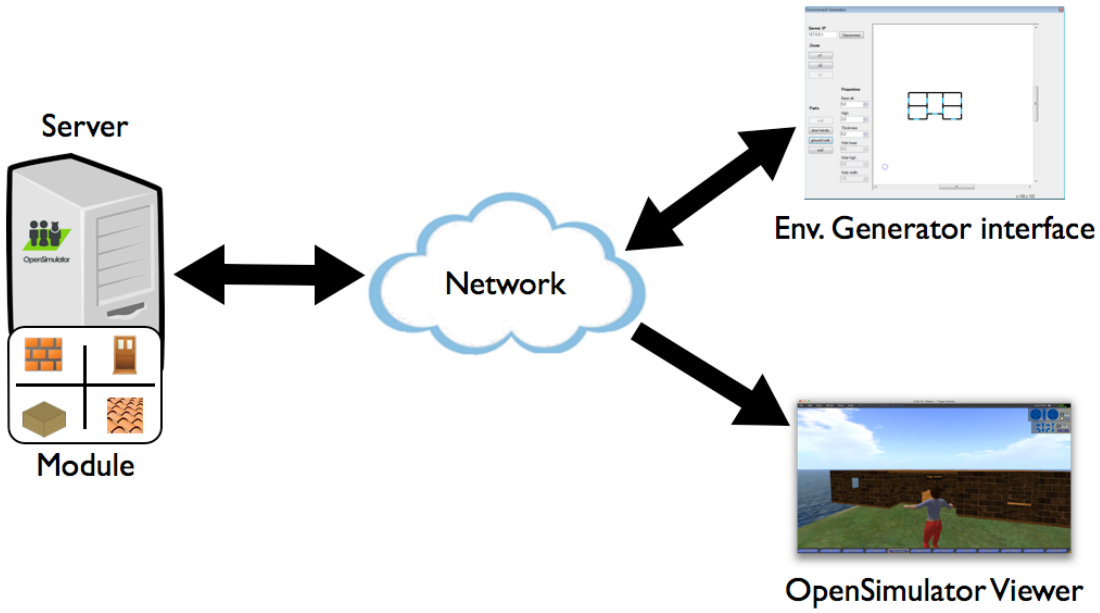


Figure 5.1: Remote access architecture

- connect - This is the first message issued. It refers to the connection establishment and no parameters are required. It must be sent to the main thread of the region module that is listening at port 12000. After establishing the connection the module assigns a private port for the new client. All the following messages must be sent through this private port, which is sent in the response of the connect message.

- quit - This is the last message a client can issue. After the server receives this message, it closes the socket listening at the private port and the connection is destroyed. No parameters are required for this type of message.

- d - This is the message that refers to the objects' deletion. It takes a parameter that is the key associated with the object that must be deleted. When this message is issued, the module searches for the received key on the list of current objects and if it exists, the object associated with it is deleted from the environment.

- w - This is the message associated with the wall kind of parts. It takes eight parameters. The first four parameters refer to the X and Y coordinates of the start and end point respectively. The next three refer to the wall base altitude, height and thickness in the OpenSimulator measures. The last one is the key string that must be associated with the wall.

- h - This is the message associated with the hole kind of parts. It takes eleven parameters. The first four parameters refer to the X and Y coordinates of the start and end point respectively. The next three refer to the base altitude, height and thickness of the wall that contains the hole. The following three parameters are associated with the hole base altitude in the containing wall, the hole height and width respectively. The last one is the key string that must be associated with the hole object that is being created.

- p - This is the message associated with the plan kind of parts. It takes seven parameters. The first four parameters refer to the X and Y coordinates of the start and end point respectively. The next two are the plan base altitude and thickness respectively. The last one is the key string that must be associated with the plan.

- r - Finally, this is the message associated with the roof kind of parts. It also takes seven parameters. The first four parameters refer to the X and Y coordinates of the start and end point respectively. The next two are the roof base altitude and height respectively. The height parameter is the dimension

between the base altitude and the peak of the roof. The last one is the key string that must be associated with the roof object.

The communication dialect provides all the functionalities of the region module through a client-server socket connection. Moreover, to ease the selection of the OpenSimulator server instance, the user interface of the solution was also lightly changed. It has now one more panel placed at the top left corner of the window as it can be seen in Figure 5.2. The connection panel is composed of a text field and a button. The text field takes the Internet Protocol (IP) address of the OpenSimulator server instance. If no address is typed in the text field it defaults for the localhost address. The button in the connection panel can show two different labels. It shows "Connect" when the interface is not connected to any server instance and shows "Disconnect" when it is connected to a valid OpenSimulator server. When the button is clicked, the environment generator tries to connect to an OpenSimulator instance using the IP address provided by the text field. If the button is clicked when the environment generator interface is already connected to a server, it will trigger a quit message disconnecting from respective server.

5.2.2 Multi-user

The APEX framework can benefit greatly from cooperative development. Despite the environment generator being a tool that streamlines virtual environment development, its utility would be limited if it only allowed one user at a time. There are very large scenarios that can be designed more quickly with cooperative work.

To provide multiple access for the current solution we might be led to think of using a region module for each user. However, this solution would not be dynamic, it would always be limited to the number of module copies we place at the OpenSimulator server folder. Instead, the desired solution can be listening at as many ports as the number of users connected, launching a thread for each one and triggering the required actions on a main thread. This solution was implemented for the environment generator and Figure 5.3 illustrates its architecture.

The region module launches a server socket when it is loaded. This is the main link and is running infinitely waiting for new user connections. It maintains a counter of the connections which is used to determine the port to assign to a

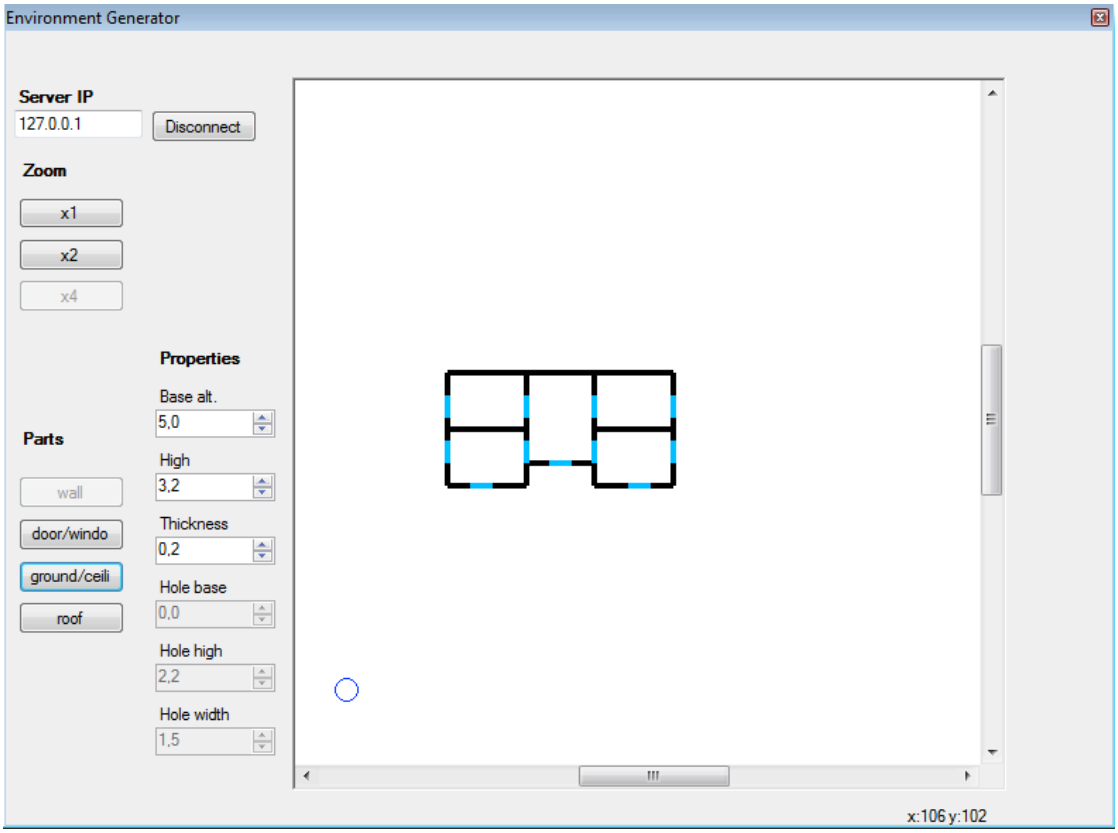


Figure 5.2: Environment generator connection panel

particular user. After receiving a new connection request, it calculates the port and launches a new thread aimed to talk with the new client at the assigned port. After this, the main thread responds to the user request with the assigned port number so that the client can continue to interact with the region module trough this new channel.

Thus, every client is answered independently but in the same region module at the same time. The client object keys and requests are not mixed since they are maintained in the respective thread memory. Moreover, the number of simultaneous active clients is not limited because the sever resources are dynamically allocated for each user connection and deallocated on each disconnection. The only inconvenient is that users can only edit their own objects. However this can be improved in the future by downloading the current objects list while connecting to the module.

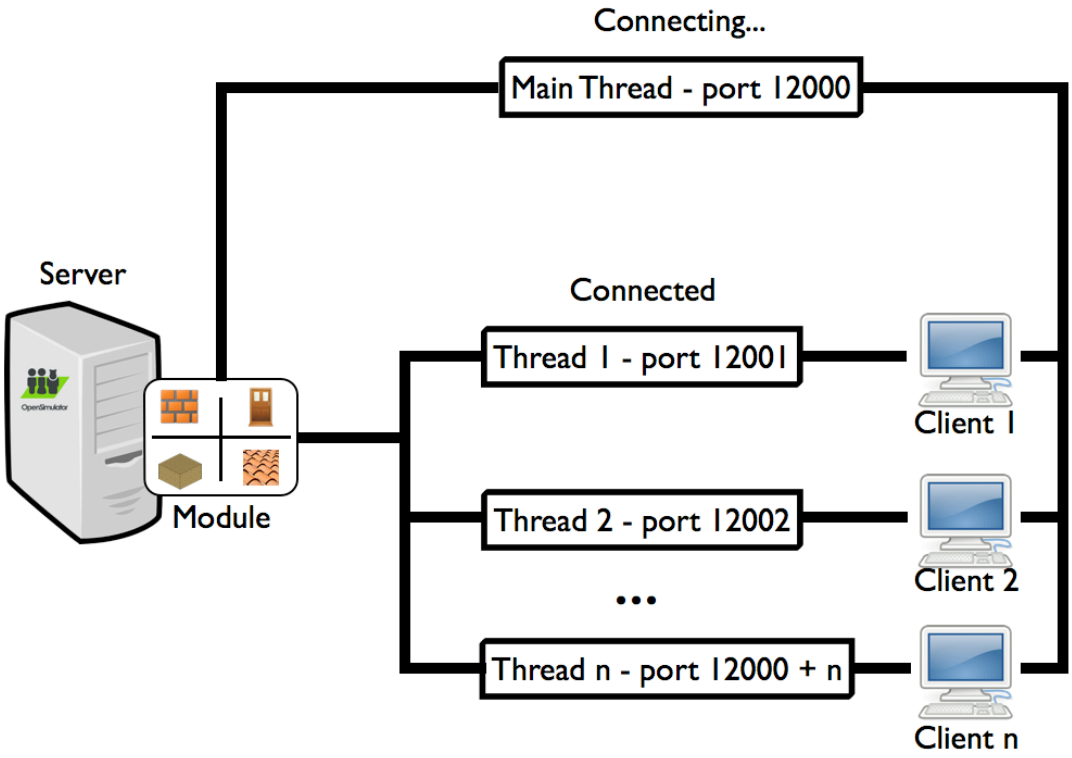


Figure 5.3: Multi-user architecture

5.3 Summary

In this chapter, the changes made to the environment generator tool were described and justified. We saw that those changes were required for the solution to be integrated with the APEX framework due to the platform's typical context of use.

In short the environment generator tool is now fully integrated with the APEX framework. In addition to the features listed, it now can be used through a remote connection and also by multiple users at the same time. Allowing remote and also cooperative development which can widely contribute to the streamlining of the environment generation process that is the main objective of this research.

The solution's structure changed a little and the architecture is now more robust, however it is more complex too. The overall sequence of actions for the creation of a building part between the user and the other components that take

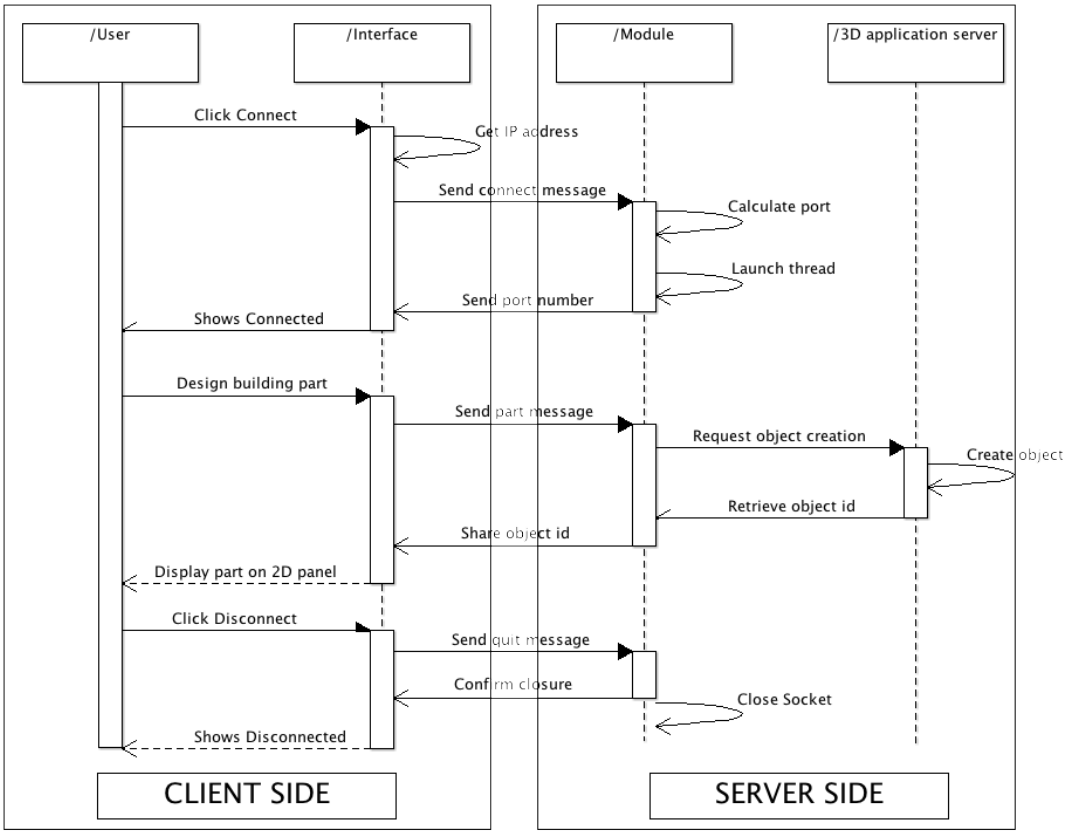


Figure 5.4: Complete solution sequence diagram

action on the solution is represented in the Figure 5.4 in the form of an Unified Modelling Language (UML) sequence diagram.

Now the environment generation solution is fully integrated with the APEX framework and ready to be used. In the next chapter an example of use of the APEX framework will be described and the virtual environment will also be tested.

Chapter 6

Developing serious games with APEX framework

6.1 Introduction

In the last chapter we saw that the environment generator solution needed some improvements in order to be integrated with the APEX framework context. After implementing those changes, the solution was fully integrated. J. L. Silva in his thesis [Sil12] proposes the use of the APEX framework to develop serious games. This idea will be explored in this chapter. A serious game is going to be developed and tested using the APEX framework where the new environment generation solution is used and compared with the previous one.

Serious games combine playing with learning. They stem from the realisation that games can be used to educate and train, as well as to offer play. Mike Zyda [Zyd05] defines a Serious Game as a mental competition, played with a computer in accordance with specific rules, that uses entertainment to promote training, education, health, public policy and strategic communications objectives. The use of games for such purposes, however, long predates the popularisation of computer gaming [Abt70].

Health education is one area where the use of serious games is being explored [BBTB08, TBB+10]. This is especially true for young people, and when using virtual environments [MFME12, BHW07] to create first person games. In first

person games, users typically control an avatar that is placed inside a 3D virtual world.

Considerable work is being carried out in exploring how to best design these games (see, for example, [BBTB08, TBB+10, MFME12, BHW07]). This chapter takes an engineering perspective, and explores the use of the APEX [SORF+10] platform for the rapid development of first person serious games. The aim is to make use of the facilities that the platform provides, in terms of creation of virtual environments and the definition of behaviours in these environments, to support the expeditious development of games.

The game, to be described, addresses the problems faced by children with asthma. Asthma is a chronic disease and specific procedures prevent the emergence of crises. The goal is to convey knowledge about these procedures to relevant children at elementary school level. The main objective of the study described in this chapter is to validate the use of the APEX platform to support the development of serious games. The success of this experience and the lessons learned from it are described herein.

6.2 Asthma

Asthma [AS02] is a chronic inflammatory disease of the respiratory tract. The most common symptoms include wheezing, coughing, chest tightness and shortness of breath. Asthma is a hereditary disease. In most detected cases, there is already a family history of respiratory illnesses.

Asthma attacks can happen for a number of reasons. The most common are drug intake during feeding or medication and inhalation of certain substances, such as pollen, smoke, animal detritus or dust. Most substances that cause asthma attacks are directly related to the existence of abundant mites (see Figure 6.1) that are very often in our homes. Objects like fabric upholstery, curtains or clothes, can build large communities of mites and cause unwanted reactions in individuals with some kind of respiratory illness. There are several procedures to avoid asthma attacks, but these procedures are not always known by asthma sufferers.

Parents and, especially, children need support to identify the causes of asthma attacks and how to avoid them. Governmental and non-governmental institutions

Figure 6.1: Mites in our homes

have developed lists of tasks (cf. [EPA04]) to instruct people about how they should proceed when faced with the problem of asthma, but these lists are not the most appropriate way to encourage children to learn how to fight asthma.

6.3 The Virtual Environment

The virtual environment for the asthma game is based in a house (see Figure 6.2) inspired by the Aware Home at the Georgia Institute of Technology [KPJ+08]. Two houses were built for further analysis. One using an OpenSimulator compatible viewer and the other one using the environment generator. The resulting environments were identical, however the building processes were pretty different. The one using the compatible viewer was built using the common process of the OpenSimulator. One primitive is created at a time. So the house structure took about 8 hours to complete. The other house was built using the environment generator. House parts like walls, windows or doors holes were quickly created using the environment generator features. However some other parts took very much time to create because there were no automatic processes for creating them.

Figure 6.2: House of the asthma game

Some examples of those parts are the stairs, window frames or automatic doors. They could be found at online libraries, however it would be difficult to adapt to the current scenario. Those parts took almost 70% of the total development time. This version of the house took about 2 hours to complete. Hence, despite being faster, it could be even more quick if the environment generator could generate some more parts automatically.

When the structure was complete, the two houses were furnished as a typical dwelling, using libraries available online (e.g., the Google 3D Warehouse[1]). Furnishings included those related to some of the main causes of asthma attacks. In this way players are able to associate asthma causing agents present in the virtual environment with real situations that happen in their homes.

6.4 The game

The game aims to convey to players some of the basic steps to take at home to prevent the causes of asthma attacks. Immersive environments allow users to experience everyday situations. They provide a textured context for identifying

[1] `http://sketchup.google.com/3dwarehouse/` (July, 2013)

the correct decisions to be taken when asthma is triggered.

Figure 6.3: Pets in the bedroom

A total of 9 objects, potentially causing asthma attacks, were placed in the house:

- domestic animals (see Figure 6.3)
- laundry abandoned on the floor (see Figure 6.4)
- fireplaces
- plush animal toys
- mouldy walls (moisture)
- curtains
- blankets
- cleansing products
- carpets

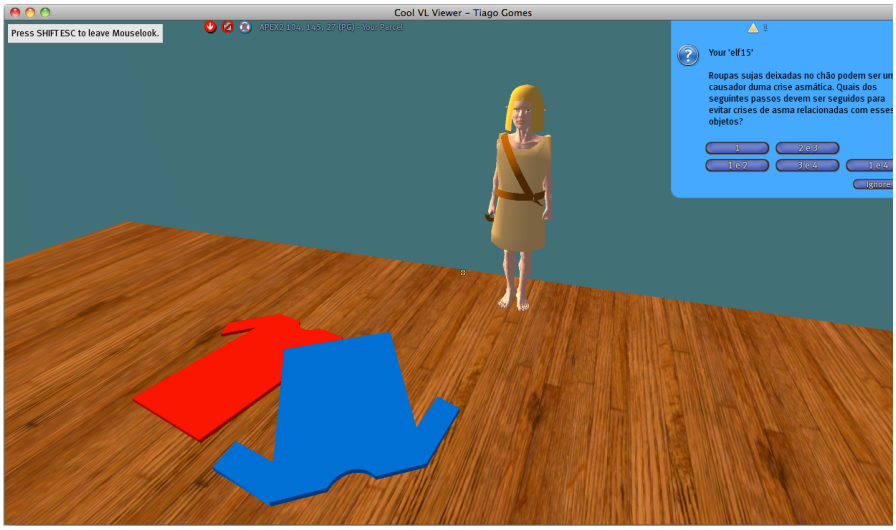


Figure 6.4: Question about dirty clothes

A character was identified with each trigger (see Figures 6.3 and 6.4) that is designed to facilitate learning. These characters provide relevant information and ask questions to be answered by the player that are relevant asthma trigger. For example, issues related to cleansing products and laundry are:

- "Cleansing products with intense odours, such as those often used in the cleaning of toilets, can cause asthma attacks. Which of the following steps should be followed to prevent asthma attacks related to these products?"
- "Dirty clothes left on the floor can be a cause of an asthma attack. Which of the following steps should be followed to prevent asthma attacks related to these objects?"

These questions test an understanding of how to proceed to avoid the asthma trigger in question. For each question players are presented with four possible answers. Of these, only two at most are correct. The player must identify the correct answers from a set of alternatives. The selection is made by pressing the button corresponding to the desired alternative.

As an illustration, the informational text, the question and the possible answers for the avatar situated next to a pet, are as follows.

Domestic Animals

Pets, such as dogs and cats, can trigger asthma attacks. Which of the following actions should be taken to avoid asthma episodes related to animals?

1. Keep those animals inside the home.
2. If possible, keep the animals out of the home.
3. Let the animals wonder freely in the home.
4. If keeping the animals outside is not possible, at least keep them away from where asthma sufferer sleeps.

Answers:

- 1 and 3
- 2 and 4
- 1
- 4

For each correct answer, the player gets a word which at the end of the game can be used to form a sentence about asthma. This is intended as an incentive for players to attempt to answer all questions.

Each player controls an avatar in the virtual world, and is allowed to attempt answering question until the right answer is found. Once all answers have been answered, the player is notified that the game has ended and told how many wrong answers have been given.

The logic of the game is implemented through a combination of Linden Scripting Language (LSL) scripts in the environment and CPN models in the behavioural component. The flexibility provided by the platform, at this level, means that different versions of the game can be generated easily. This includes changing the game logic, the number and type of asthma triggers present, and the questions.

6.5 Evaluation

A user study was carried out to evaluate the ability of virtual environments such as the one just described to work as serious games to promote learning. The target

audience were children aged 9 to 10. Hence, the study was set up to mainly address questions related to the usability of the environments and the satisfaction of users within them.

6.5.1 The user study

The target audience for the study were children aged 9 to 10, attending the fourth year of the first level of studies (in Portugal). All children possess a laptop able to run the necessary software to interact with the virtual environment. Their computers were therefore used for the study.

Besides logging the user actions in the environment, a questionnaire was used to obtain information about the users (age, gender, previous experience with similar games, and previous knowledge of respiratory diseases). This was used to evaluate the utility and ease of use of the game, as well as to obtain information about perception of learning by the users. The developed questionnaire is available in Annex A.1. Because of the age range of the target audience, a simplified 3 point Likert scale was used.

Prior to the study all machines were prepared for using the environment. An OpenSimulator compatible viewer (Cool VL Viewer) was pre-installed and configured in a total of 18 machines. These enabled access to the virtual environment. At the start of the study, participants were provided with information and instructions about how to use the platform without problems. This included some time to use the Cool VL Viewer and the virtual environment. Following that, all subjects had 30 minutes to play the Asthma game, trying to answer all questions. During the experiment three evaluators were present which, besides solving a few technical problems with the machines, mostly observed the users during game play.

After the game period, each player answered the questionnaire mentioned above. The data collected helped not only to understand whether the APEX framework can be used to create serious games, but also to evaluate the actual game that was developed, identifying possible shortcomings and improvements.

6.5.2 Results

The analysis involved 18 children as participants (11 males and 7 females). None of the children had previous experience with 3D application servers (e.g. Second Life), but 12 stated that they had played computer games before.

Children's reaction to the game was quite positive. According to the data collected from the questionnaires, out of the 18 children, 16 found the game fun to play, 12 found it easy to play, and 15 said they had learnt something about respiratory diseases. None answered negatively any of the questions (see Figure 6.5).

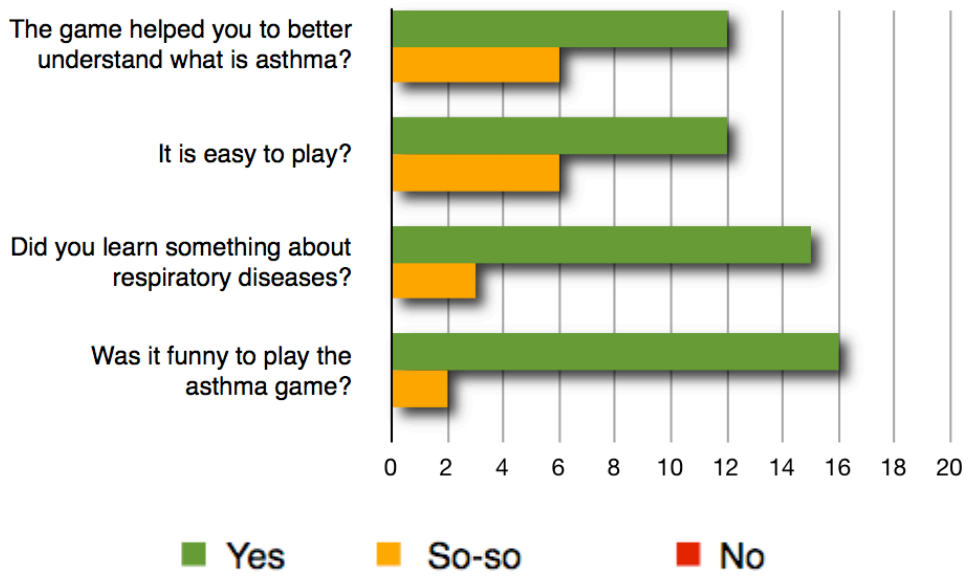


Figure 6.5: Survey results

Regarding utility, the collected data shows positive results. Twelve children stated that they were able to better understand what asthma is after playing the game. The same number of children felt that, after playing the game, they were better prepared to help people with respiratory problems. Fourteen stated that they were prepared to act in their homes to avoid such problems. They all stated that they considered applying what they had learnt in their homes. The results of this section of the questionnaire can be found in Figure 6.6.

In terms of learning, only 2 of the children that played the game found the

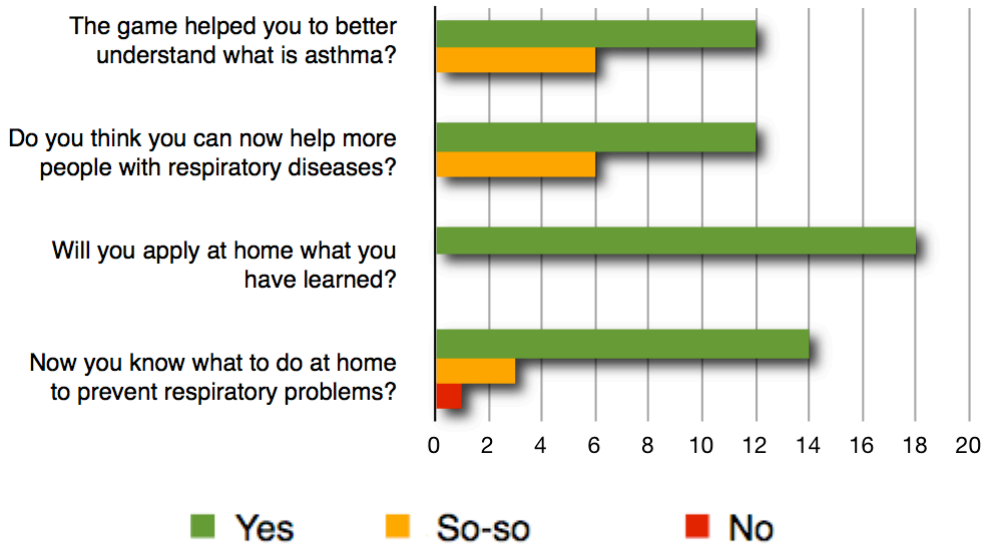


Figure 6.6: Utility section results

questions hard. This indicated that the game is accessible and can be played by young players.

In terms of satisfaction, there were also positive results. Fourteen players said they would recommend the game to friends. It should be noted that none of the others responded negatively to the question (they gave neutral replies).

Despite these results, none of the players was able to finish the game in the allotted 30 minutes slot. It should be noted that some players kept playing for a while (at school) after filling in the questionnaire, and also after going home. From the observations during game play, it was concluded that certain features of the virtual environment, as well as access to some of the configuration options of the virtual environments server, contributed to some degree of distraction during game play. In fact, a considerable number of children, felt more interested in exploring the environment, and interacting with the other players in the environment (chatting, pushing other avatars), than in trying to finish the game by answering the questions.

More specifically, the main factors that were found to distract children from the actual purpose of the game were the following:

1. It could be observed that one of the main distracting factors was the possibility of going into the sea that surrounded the island were the game was set, in order to explore the sea bottom.

2. The fact that avatars could fly was another influential distraction during game play.

3. Another aspect that reduced the focus on game goal was the ability to create new objects in the virtual world, as well as that of changing already existing ones.

4. Finally, the chat feature present in OpenSimulator also contributed to some distraction.

6.6 Game redesign

As mentioned above, during the study it was found that some of the props used in the virtual environment, as well as some of the aspects inherent to virtual server environments used, contributed to lack of focus on the game by the players. This was detrimental to the ability of the players to reach the end of the game. Thus, in order to better focus users on the game aspect of the environment, thus better promoting learning about prevention of the symptoms caused by asthma, a redesign of the game's environment was carried out. A second version of the environment was created that addressed the shortcomings found at this level during the study.

Regarding the environment, it was found that a major distracting factor was the fact that the entire environment was surrounded by water, which the users could enter and explore. To avoid this problem, in the second version of the environment a transparent boundary was created between land and water (see Figure 6.7). This served to restrict the playing area to the ground zone. However users would still have the possibility of entering the water area by activating the flight mode and flying over these obstacles. Since it had also been established that the flight feature was another major distracting factor, in this second version of the environment this feature was disabled. With both these changes implemented

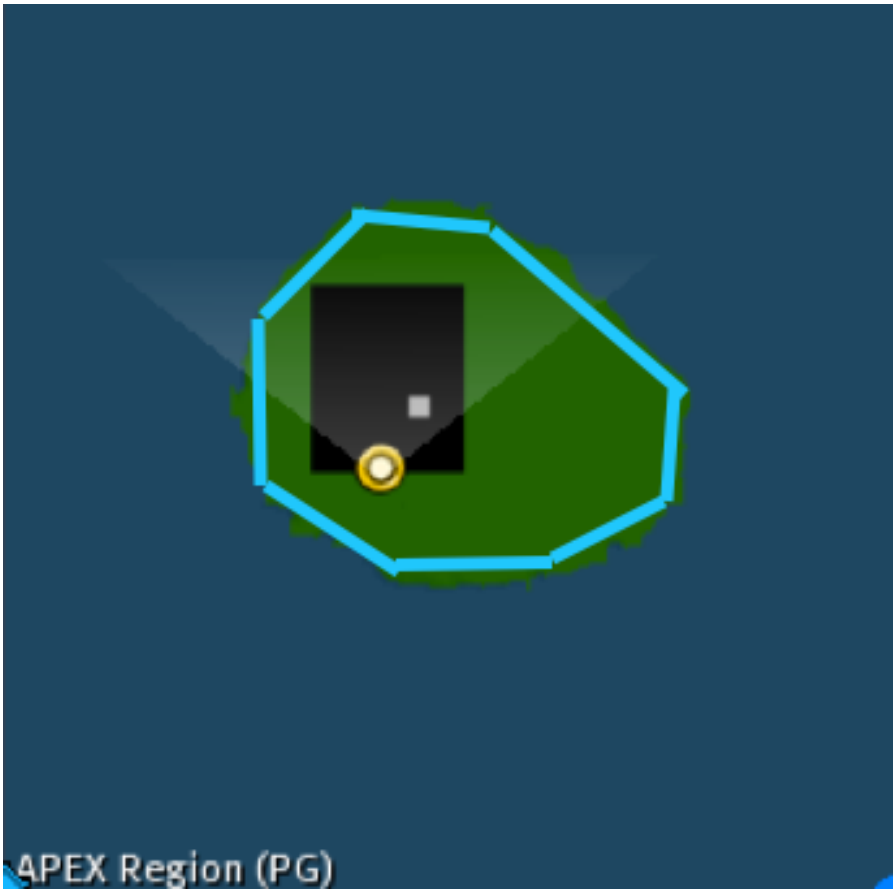


Figure 6.7: Useful area bounded by barriers

both the entry into water and the distraction caused by the functionality of flight were avoided.

Another aspect which decreased the focus on the game was the ability to create new objects in the virtual world, as well as to change or remove the existing ones. In the initial study, a significant number of players lost a good percentage of the time with these features to change the environment used. To avoid these distracting factors, the features of construction, and edition of the environment were blocked in the second version of the game environment for all users with normal privileges.

The chat instant messaging was also one of OpenSimulator's features which contributed to the lack of focus in the first study. The chat functionality, however, could not be disabled in the second version of the environment as it is used during the game to start counting correct and incorrect responses for each player.

Due to the high adherence of the players after the first use, we felt the need to record all accesses made subsequently by users. This type of information may prove useful for future analyses on the use of the platform. To achieve this a record of accesses was created in the second version of the environment, which keep the user identification and a temporal label for each access.

With these changes already made, a second edition of the study is currently being prepared.

6.7 Conclusions

Serious Games aim to combine learning with entertainment. Health education is one of the areas in which this approach has already proved useful, particularly in the case of young people, and when virtual environments are used. In the context of the APEX project an approach was developed to the rapid development of simulations of ubiquitous computing environments. In this chapter we explored the use of this approach, together with the environments generation tool, for the development of a serious game.

To demonstrate the feasibility of the approach we developed a game that addresses the problems faced by children with asthma. Two houses were built for the game. One of them using an OpenSimulator compatible viewer and the other one using the environment generator. Building the structure of the house using the viewer took about 8 hours, while using the environment generator took about 2 hours. However the latter could have been quicker if, for example, the environment generator could generate the stairs too. The stairs of the house were the slowest part of the building development. Here, only the structure development time was considered because almost every other objects, like furniture, were imported from on-line 3D libraries. Although a more thorough study needs to be carried out in order to validate the generator tool developed, this results clearly indicate the good potential of the tool to streamline the development of environments in the context of APEX.

The intention of the game is to impart knowledge on how to act when faced with factors that might cause asthma attacks, to avoid these same attacks. In order to validate the concept, a user study was conducted. Through this study

it was revealed that although the virtual environment has captured the attention of children, there is a need to restrict what avatars can do in the environment, in order to better focus the players on the goal of the game.

A second edition of the study is currently being prepared, with a new version of the game. It is, however, possible to say at the outset that the APEX framework presents itself as a promising approach for the rapid development of serious games.

Chapter 7

Conclusions and future work

This dissertation, developed in the APEX framework context, documented the research for an agile solution for generating 3D virtual environments with a view on the prototyping of ubiquitous environments. The research work resulted in three published papers [GASC13, GAH+13, AGSC13].

In this chapter, an overall analysis of the work done is made. Also, an analysis of the objectives' completion is discussed according to the solutions found for the dissertation problem. And finally, a proposal for future work is made taking into account the limitations of the solution found and upcoming experiences using the APEX framework.

7.1 Overall analysis

It was stated in the first chapter that the dissertation's main objective was to build a component for the APEX framework supporting the rapid generation of 3D virtual environments. It was also said that an important aspect was the accuracy of the 3D models generated with the developed tool. Moreover it was declared that APEX users require that the environment can be developed incrementally. According to these requirements we can affirm that the objective of the research was achieved. More than that, we are able to say that the targets were overachieved. In addition to overcome the proposed requirements, the solution presented also provides remote access and multi-user development.

In Chapter 2 the relevant topics for this work were studied. The chapter started by describing the APEX framework and its environment component which is composed of a 3D application server, the OpenSimulator, and a compatible viewer. Still about OpenSimulator, a description of its API was made. Those were really important topics since they took part on some of the solutions presented afterwards. The OpenSimulator API for creating region modules was also documented.

Then, in Chapter 3 OpenSimulator region ARchives (OARs) and relevant virtual environment modelling languages and techniques were described. This knowledge helped to understand how virtual scenarios are structured, how we might interact with them and which technique can be used for each type of scenario. Still in this section COLLADA files were addressed, the standard files for uploading 3D objects into the OpenSimulator virtual environment.

Afterwords an analysis of some useful 3D modelling tools that could be integrated to develop solutions for the environment generation problem was made. Only open source tools were analysed so that we could easily access their source code and tune them for the problem purposes. Also, one of these tools, Sweet Home 3D (SH3D), introduced a new concept for user interfaces targeted to building development. The transformation of a 2D panel into a 3D scenario makes it simple and agile to develop virtual environments.

At the end of Chapter 3, a discussion about the 3D modelling tools that could be used to build a solution for the problem took place. Some of the current approaches were eliminated due to lack of some of the problem required features. Some other were elected in the chapter conclusions to make part of the solutions that were analysed in the next chapter.

Chapter 4 documented the development and testing of the possible solutions for the virtual environment generation problem in the APEX framework. A building model was defined at the beginning of the chapter so that every solution developed could be fairly compared. Then, two main approaches for developing solutions were described. One was based on the use of 3D modelling tools to compose solutions that can output OpenSimulator readable models (COLLADA). The other approach make use of the OpenSimulator API and some knowledge about user interfaces targeted for building development to compose a solution.

Solutions created from the first approach showed ease of use and great accuracy. Although the solution using the OpenSimulator API was more dynamic, more independent from external entities and the most important, it was almost twice as fast in the building model test. Moreover these solution benefits from real time creation of the virtual environments. Hence, in the end of Chapter 4 we concluded and choose the OpenSimulator API solution for the virtual environment generation problem at APEX framework. However, we stated that the current solution required some changes in order to be fully integrated with APEX.

Chapter 5 started introducing the environment generation solution features that were required for it to integrate with the APEX framework. We saw that the default context of use of the framework required some more about the solution. It should be able to be accessed remotely because APEX users rarely use the framework in a local context. And it should also be able to be used by multiple users simultaneously so that large scenarios could be developed with cooperative work. The solution's architecture changed a little, first a network layer was inserted between the region module that was placed at the server and the user interface that was placed at the client side. These layer required the creation of a communication dialect so that the module features could be called remotely. Also, a connection panel was added to the interface in order to connect it to the desired server instance.

To provide multi-user capabilities some changes were also made. There is a main communication channel that receives all the connection intents, and then a new channel thread is created for each new client so that everyone is answered independently.

At the end of Chapter 5 we concluded that the environment generator was fully integrated with the APEX framework and ready to use. So, the next chapter documented the application of the APEX framework for building a serious game.

The game created in Chapter 6 is about respiratory diseases, more concretely about asthma. The chapter started describing the importance of serious games, and a brief description of asthma was made. Then the asthma game development was documented from the environment creation to the game logic. After describing the game a study with children was presented as an evaluation. The study emphasises the APEX framework capabilities. Although, the game showed that it

could be more objective while the study occurred. Thus, the game was remodelled and prepared for another experience. The game redesign is also documented in Chapter 6.

The chapter finishes concluding that the APEX framework can be easily used for developing serious games, although we must be careful with the environment distractors so that they cannot contribute for the players distraction. The second iteration of the study is being prepared. It will use the improved version of the asthma game.

The APEX framework is now richer than before. Its ability to generate virtual environments improved a lot with this work, and the environment generation tool is also more focused in the framework objectives. However the environment generator can be improved, as it was said in Chapter 6, by automating the creation of more building parts, like stairs, window frames or even round walls.

7.2 Results

The main objective of this project was to develop a component for the APEX framework to streamline the generation of 3D virtual environments. This component aims to optimise the prototyping of ubiquitous environments speeding the development of the scenario and so giving more time to concentrate with the ubiquitous systems. Thus, making the whole process more efficient.

The component developed should show a high level of accuracy of the generated objects because it is an important factor for the final users experience of the environment. The users must easily recognise the scenario retreated so that the prototyping process can be valid and efficient. Also, the component must allow incremental building of the environment. APEX users frequently need to change the environment, either to insert new objects or remove the existing ones. Hence the environment generator may be able to do this.

Below it is described how each of these features was achieved:

- The component developed for the APEX framework that streamlines the virtual environment generation process is the solution using the OpenSimulator API chosen on Chapter 4. The developed component shows high potential

on its approach to develop the virtual worlds. The transformation of the objects designed in a 2D panel into 3D objects in the virtual environment is a very powerful technique that confers agility to the process. Moreover, all the designed objects can be seen in real time on the virtual world through an OpenSimulator compatible viewer.

- The accuracy of the objects created using the environment generator is quite acceptable. The 3D objects created are OpenSimulator native objects (prims). All the objects created through the environment generator interface are dimensioned in the OpenSimulator measures. Also the properties panel on the interface allows the adjustment of the objects sizes down to the decimetre. The objects created using the environment generator can be seen in Chapter 4 where a building was modelled using it.

- To provide APEX users with an incremental process of environment creation some further features were taken into account. One important aspect to achieve this objective was to build the objects created with the environment generator in real time, so that users could see the result of their actions on every step. The delete feature implemented in the environment generator is also a very important one. This allows users to easily remove unwanted objects or errors that occurred while designing. Maybe the most important aspect to achieve this objective was the fact that the chosen solution does not require erasing any of the objects already created to change the environment. The user can freely add new objects at any time without affecting the existent ones.

Beyond the objectives of the research there were also some features that helped to assert the environment generator efficiency and convenience in the APEX framework context. Those features were documented in Chapter 5 on the integration of the environment generator with the APEX framework.

The remote access feature confers convenience to the tool. It would be very inefficient to move to the OpenSimulator server machine every time the environment generator had to be used or even to maintain a server instance at every machine just to have access to the environment generator features.

The multi-user access feature confers not only convenience but also efficiency to the framework. It is a nice aspect not to be limited to an user at a time, but the most important is that it is possible, with this feature, to cooperate with several users in the development of a large environment for example. This makes the framework more agile too.

To sum up, the objectives of this research project were successfully achieved. Every item was taken into account to build the solution and also some improvements were added besides the main objectives. However, some work has to be made in order to make the framework more mature, more effective and to prove the framework capacities too. The future work is described in the section below.

7.3 Future Work

The developed solution achieved the objectives of the dissertation, although it can be greatly improved. The fact that the chosen solution was developed from scratch makes it easy to add new features or improve the existing ones.

For example, the environment generator is for now limited to four different types of building parts. Any wanted new part can be added to the tool. The tool is not able to build stairs, unless we build them using the existent structures. However, building stairs could be an automated action joining two floors.

Another important aspect that could be improved is the choice of the objects' texture. For now the objects are created with a default texture, which can be changed later, but it would be very convenient if we could choose the texture for each part we design as it is done with the object dimensions for example.

The existent features could also be improved. For example, when a door is created, a door object could appear automatically with a LSL script associated to it so that it rotates when an avatar approaches as a real door does. The windows could also be improved. Instead of making just a hole in the wall, a glass object could also appear in the middle of the hole, thus simulating a real window.

The existent building parts defined by rectangles (plans and roofs) could be improved too. Instead of being defined by rectangles they could be defined by free lines if the user wanted it. Hence, this improvement would make these parts' design more flexible.

To prove the framework capacities, some work can be done too. As it was said in the end of the fifth chapter, the asthma game was redesigned and improved in order to become more focused in the game objectives. A second edition of the study documented in this chapter is being prepared at the moment. This experience will help to find more limitations and not so good aspects of the framework and contribute to its development and growth.

Appendices

A.1 Questionnaire

The questionnaire used in the users study to evaluate the Asthma Game is presented below (translated from the Portuguese original).

Subject characterization				
1	Age			
2	Gender			
		No	**So-so**	**Yes**
		☹	😐	☺
3	Do you often play computer games?			
4	Have you ever played a game like this?			
5	Do you know OpenSimulator or SecondLife?			
6	Do you already know any respiratory disease?			
Utility				
7	The game helped you to better understand what is asthma?			
8	Do you think you can now help more people with respiratory diseases?			

9	Will you apply at home what you have learned?			
10	Now you know what to do at home to prevent respiratory probl?			
Ease of use				
11	Is it easy to play?			
12	Is it simple to walk through the game's house?			
13	Is it possible to play without previous instructions?			
14	Is it more easy to reply to those questions or to the game questions?			
Learning				
15	Were the questions difficult?			
16	Did you learn something about respiratory diseases?			
17	If you have to make a test about respiratory diseases, would it be more easy now?			
18	Can you explain how to proceed in order to prevent asthma attacks?			
Satisfaction				
19	Are asthma triggers well represented in the game?			
20	Was it funny to play the asthma game?			
21	Is it tempting, to try to figure out all the questions to complete the game?			
22	Have you easily identified the moments when the game started and when it finished?			
23	Would you recommend the game?			

Bibliography

[AB06] Remi Arnaud and Mark Barnes. *Collada: Sailing the Gulf of 3d Digital Content Creation*. AK Peters Ltd, 2006.

[Abt70] Clark Abt. *Serious games*. The Viking Press, 1970.

[AGSC13] Tiago Abade, Tiago Gomes, José Silva, and José Campos. Concepção e Avaliação de Ambientes Ub quos na Plataforma APEX. In *Interacção, Vila Real - Portugal, 07/11/2013-08/11/2013*, 2013.

[AS02] Lara Akinbami and Kenneth Schoendorf. Trends in childhood asthma: Prevalence, health care utilization, and mortality. *Pediatrics*, 110(2):315–322, August 2002.

[BBTB08] Tom Baranowski, Richard Buday, Debbe Thompson, and Janice Baranowski. Playing for real: Video games and stories for health-related behavior change. *American Journal of Preventive Medicine*, 34(1):74–82, 2008.

[BHW07] Maged Boulos, Lee Hetherington, and Steve Wheeler. Second Life: an overview of the potential of 3-D virtual worlds in medical and health education. *Health Information & Libraries Journal*, 24(4):233–245, 2007.

[CBSF07] António Coelho, Maximino Bessa, António Sousa, and Fernando Ferreira. Expeditious modelling of virtual urban environments with geospatial l-systems. *Computer Graphics Forum*, 26(4):769–782, 2007.

[CCC+08] Paolo Cignoni, Marco Callieri, Massimiliano Corsini, Matteo Dellepiane, Fabio Ganovelli, and Guido Ranzuglia. Meshlab: an open-source mesh processing tool. In *Sixth Eurographics Italian Chapter Conference*, pages 129–136, 2008.

[CCR08] Paolo Cignoni, Massimiliano Corsini, and Guido Ranzuglia. Meshlab: an open-source 3d mesh processing system. *ERCIM News*, 2008(73):45–46, April 2008.

[EPA04] US EPA. Asthma Home Environment Checklist, United States Environmental Protection Agency, 2004.

[EV01] Rae Earnshaw and John Vince, editors. *Digital Content Creation*. Springer, 2001.

[GAH+13] Tiago Gomes, Tiago Abade, Michael Harrison, José Silva, and José Campos. Developing Serious Games With The APEX Framework. In *Ubiquitous games and gamification for promoting behavior change and wellbeing, Trento - Italy, 16/09/2013-16/09/2013*, 2013.

[GASC13] Tiago Gomes, Tiago Abade, José Silva, and José Campos. Desenvolvimento de Jogos Educativos na plataforma APEX: O Jogo da Asma. In *Interacção, Vila Real - Portugal, 07/11/2013-08/11/2013*, 2013.

[Jus08] Justincc. Opensim tech basics: Oars - opensim region archives. `http://justincc.org/blog/2008/10/10/opensim-tech-basics-oars-opensim-region-archives`, October 2008.

[Jus09] Justincc. A little bit more on oar. `http://justincc.org/blog/2009/05/01/a-little-bit-more-on-oar/`, May 2009.

[KPJ+08] Julie Kientz, Shwetak Patel, Brian Jones, Ed Price, Elizabeth Mynatt, and Gregory Abowd. The georgia tech aware home. In *CHI '08 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '08, pages 3675–3680, New York, NY, USA, 2008. ACM.

[MFME12] Houda Mouaheb, Ahmed Fahli, Mohammed Moussetad, and Said Eljamali. The serious game: What educational benefits? *Procedia - Social and Behavioral Sciences*, 46:5502 – 5508, 2012.

[Pen84] Alex Pentland. Fractal-based description of natural scenes. Technical Report 280, AI Center, SRI International, 333 Ravenswood Ave., Menlo Park, CA 94025, Feb 1984.

[SC11] Pedro Silva and António Coelho. Procedural modeling for realistic virtual worlds development. *Journal of Virtual Worlds Research*, 4(1), 2011.

[Sil12] José Silva. *Rapid Prototyping of Ubiquitous Computing Environments*. PhD thesis, Escola de Engenharia da Universidade do Minho, March 2012.

[SORF+10] José Silva, Óscar Ribeiro, José Fernandes, José Campos, and Michael Harrison. The apex framework: prototyping of ubiquitous environments based on petri nets. In *Human-Centred Software Engineering*, volume 6409 of *Lecture Notes in Computer Science*, pages 6–21. Springer, 2010.

[Tal96] Habib Talhami. L-systems for three-dimensional anatomical modelling: Towards a virtual laboratory in anatomy. In Karl Heinz H hne and Ron Kikinis, editors, *VBC*, volume 1131 of *Lecture Notes in Computer Science*, pages 393–398. Springer, 1996.

[TBB+10] Debbe Thompson, Tom Baranowski, Richard Buday, Janice Baranowski, Victoria Thompson, Russell Jago, and Melissa Griffith. Serious video games for health: How behavioral science guided the development of a serious video game. *Simulation & Gaming*, 41(4):587–606, 2010.

[Woo03] David Woolford. Understanding and using scene graphs, June 2003.

[Zyd05] Michael Zyda. From visual simulation to virtual reality to games. *IEEE Computer*, 38(9):25–32, September 2005.