Erasmus+

Project funded by: **Erasmus+ / Key Action 2, Strategic Partnership for Vocational Education and Training.**

(European Commission, EACEA)

O1D3 - REPORT ON STATE OF THE ART IN VIRTUAL WORLD TECHNOLOGIES

Executive Summary

Research on the State of the Art in 3D Virtual Reality Worlds and frameworks is considered to be very important, regarding the work to be conducted in the scope of the project. This report aims to serve as an introduction to 3D Virtual Worlds, tools and frameworks that are available, identify the strengths and weaknesses of each one of them and select the most appropriate one to be used within the VR-WAMA project.

OBSAH

1 INTRODUCTION

A virtual world or massively multiplayer online world (MMOW) is a PC based, simulated environment in which a plethora of users can create individual avatars and all the while, freely investigate the virtual environment and partake in numerous activities in the virtual world, as well as communicate with each other.

A generally accepted definition of virtual world does not exist. However, virtual worlds do require that the world be persistent; meaning that the world must continue its existence even after a user exits the world, and also, changes made by users to the world should be preserved [2, 3]. Although the interaction with other participants is happening in real-time, time consistency is not always maintained in online virtual worlds.

Numerous 3D Virtual Worlds exist, each one of them having various purposes, ranging from socialization and leisure, to other, more formal approaches, such as commercial or educational.

1.1 Definition

"A virtual world is a digital environment (usually graphical, usually 3D) completely delivered over the Internet or Intranet, where users are represented by Avatars, interact with each other, interact with and effect their environment in a persistent manner, have no more restrictions placed on them than they can expect in the real world, can decide from a wide range of actions, or even inaction, can build and create within the world, without having to master additional tools, can use the world for a wider variety of different purposes" [2].

In literature, there are several definitions of VW, all of which have the following characteristics [3, 4]:

- The virtual environment, meaning Graphical User Interface
- Interaction with the virtual environment itself and with digital content within the virtual environment
- User avatars: human controlled digital representations that allow the interaction within the VW in real time
- Shared space between multiple users

- Real-time interaction between users
- User communication through text and/or voice
- Formation of social groups, formulated by networks of users
- Persistence: it is guaranteed that the VW, the virtual objects within the VW and the user interaction effects on the VW, will be present even after the user has left the VW
- Computers in networks (servers) that manage all the data

2 VIRTUAL WORLDS AND EDUCATION

The environments that possess the above characteristics of the 3DVWs, could be, potentially, transformed into "educational virtual environments". An "educational virtual environment" is defined as an environment that is based on a certain pedagogical model and it incorporates or implies one or more didactic and learning objectives, providing users with experiences they would otherwise not be able to experience in the physical world (or in a classroom), while achieving specific learning outcomes[5].

Therefore, a growing interest is observed in teaching and learning within 3DVWs and various schools and universities possess such VWs for educational purposes. In many of the cases in which schools and universities use educational VWs, they tend to project their facilities to the VW. So, 3D educational VWs are mostly used as virtual classrooms or safe, simulatied, environments.

In comparison with other e-learning technologies, 3DVWs use immersive 3D experiences, which allow the learner to navigate freely inside the learning environment, explore it, obtain sense of purpose, act, make mistakes, collaborate and communicate with other learners, all of them being actions that provide them with a full understanding of a situation[2]. The most important and unique features that 3DVW technologies provide are the sense of immersion, i.e. the sense of "actually being in there" and the sense of presence, i.e. the feeling that the user belongs in the virtual world as an entity that interacts with other entities, in a way similar to being present in a real physical environment.

Furthermore, the sense of presence is enhanced by the use of anthropomorphic avatars. In that way, collaborative training activities, such as games and simulations, could be enhanced in 3DVWs occupied by avatars that look like humans. Additionally, immersion and interaction with virtual objects can boost learners' interest and engagement to the learning tasks and help them to develop a better conceptual understanding, according to the content [6, 7].

However, a very interesting statement is that the simple use of highly immersive technologies alone could not be as effective as it could potentially be if it is combined with specific design strategies [8].

2.1 Virtual World Platforms

Virtual World platforms are the tools used for creating highly immersive 3D interactive online environments that can be either duplicates of existing physical places, or even imaginary and abstract places. They are used to represent places that are very difficult, or even impossible, to visit in real life because of factors that restrict the actual presence of the user, such as cost or safety.

Regarding the technological solutions for virtual worlds, the approaches that are typically used can be grouped into two categories: *use of software solutions already developed* (but still customizable) and *use of game-engines* (for the ground-up creation of the virtual world). These categories are characterized by features that may represent strengths or weaknesses, depending on the type of intervention and the educational objectives.

The VW platforms that exist can be proprietary or open-source. In the following of this section, the most important and popular proprietary and open-source VW platforms are presented.

2.2 Proprietary Platforms

2.2.1 Second Life

Second Life (SL) - (2003 by Linden Lab) is the most popular VW platform and has the largest active community.

Figure 1: Second Life

Most important Features:

- 3D graphical environment
- The users can fully customize their avatars
- Built-in voice and text communication
- A social network, groups, as well as information and object sharing are included
- Has an enormous market: users are able to create objects, goods and scripts inside the virtual world
- Registration and basic usage are free but the users can pay a monthly subscription fee for virtual land, in order to build a home and become 'residents' (however, for large building projects, the purchase of a large piece of private land - or an island is required)
- Only specific group members can access private land
- In-world virtual currency is included (Linden Dollars (L$): 2500L$ = 10.09USD)
- In the field of education, many learning organizations from all around the globe are augmenting their current curriculum with virtual learning module, or even conduct classes and educational programs in SL

Architecture:

- Server – Client
- Server: owned & hosted by Linden Lab
- Client or 'viewer': open-source, free, runs on many platforms

Technical Characteristics:

- Physics engine: Havok 7
- Fully customizable avatars
- Terrain modification
- Built-in 3D building tool
- In-world scripting: Linden Scripting Language (LSL)
- Image, video, audio file upload
- Web pages and media on virtual objects
- Groups and full user profiles
- Communication methods: Vivox, Inc. built-in voice, text chat and IM
- Region backup and upload for landowners
- In-world economy

- Restriction in access and building rights
- Head Up Displays (HUDs)

2.2.2 Active Worlds

Active Worlds - (1997) is similar to Second Life, regarding their functionality. There is a “tourist account” for free usage. Despite this, paying a small monthly fee buys a "citizenship". A "citizens" enjoys privileges, such as a unique name, unrestricted access to the VW, avatar customization, object building, access to social networking and communication features (voice chat, Instant Messaging (IM) and file sharing).

Figure 2: Active Worlds

Private firewall-protected Universes are also available, usually for enterprise and educational purposes, in order to give more control and privacy over environments. These Universes are distinct worlds from the main world, with varying cost. Another set of worlds and a community for educational projects, with over 80 organizations, exist under the name “Active Worlds Educational Universe”, as well.

Architecture:

- Server – Client
- Server: free for Windows & Linux
- Client: free only for Windows

Key Characteristics:

- Physics engine: NVIDIA PhysX
- Avatar customization
- Development of NPCs
- Library of over 6000 models

- Built-in 3D building tool
- Support modeling programs (Truespace, Blender, Studio Max, Google Sketchup, Collada, etc) & import of custom created 3D models
- Office documents display tools
- Basic scripting system (for basic object interactivity)
- Active Worlds SDK (C/C++ and Visual Basic/COM versions), for development of in-world applications
- Built-in VOIP, text chat, IM
- File transfer
- Heads-Up Display
- Ability to restrict access and building rights

2.2.3 Jibe

Jibe is a 3D VW developed by ReactionGrid Inc. An important characteristic of Jibe is that the VWs can be embedded in webpages and be accessed via mobile devices. VWs can either be hosted by ReactionGrid Inc. or deployed on private servers. The installation of the Unity web plugin is required by Jibe, while Android and iOS support is under development.

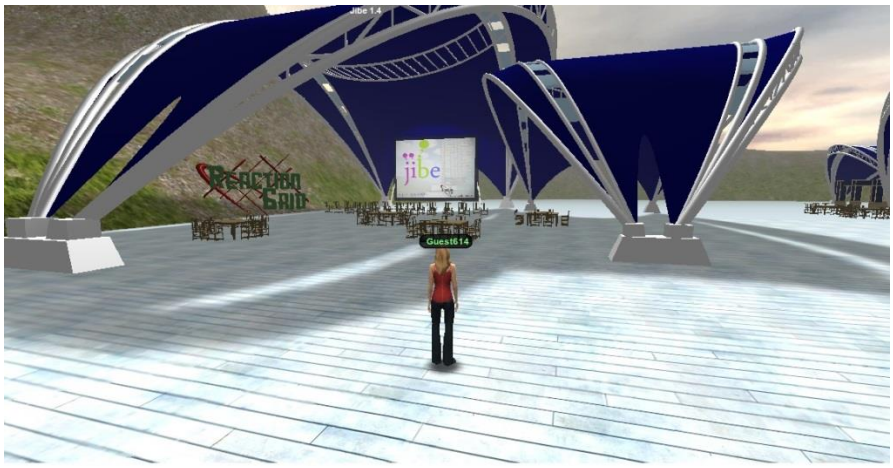

Figure 3: Jibe

In order to create a Jibe VW, the creator must use the Unity 3D editor. The result is a professional development environment characterized by professional quality graphics, physics and sound. Also, Jibe supports creating 3D objects, as well as importing 3D models from Maya, Blender or other 3D-design platforms.

Architecture:

- Backend: SmartFox Server
- Frontend: Unity 3D
- A middleware layer for the communication between backend, frontend and user database communication

Key Characteristics:

- NVIDIA PhysX Physics Engine
- Up to 100 concurrent users
- User Registration System with Database & Password Reset Forms
- In-world event tracking system - logins and in-world events are logged to a database
- Free Unity 3D editor for world creation
- Import mesh files from Sketchup, Maya and more
- Sit, teleport scripts, animations
- Dynamic Slideshow System
- Scripting in C#, JavaScript or Boo for enabling object interactivity
- Group and private chat
- Vivox Voice
- Web-plugin for browser based virtual worlds
- Windows and Mac standalone version
- Option to convert Jibe world into Android mobile app (additional charge)

2.2.4 Unity

Unity - is a 3D professional game development tool which can be used to create suitable training simulations and educational 3DVWs and not a VW platform. Access to Unity can be accomplished through a client or a web-based player.

It can be used for the development of a game, along with its user interface, without the need of programming. The only requirement in order to develop single-player games/apps is to download and install Unity. However, the features and properties of the training environment that has been developed rely mostly on the ability to use the content creation tools.

Figure 4: Unity

The Unity Asset Store is a Unity global marketplace, which provides content such as game creating tools, character models, landscape painting tools, audio effects, visual programming solutions and scripts, for free or for a low price. Unity evolves along with the latest desktop (PC, Mac, Linux), Web (web player, Flash), console (Wii U, PS3, Xbox 360) and mobile (iOS, Android) technology.

Unity is a game engine developed by Unity Technologies. It is a development environment used to create video games, cross platform simulations and movies, both in 3D and 2D. The development environment is multi-platform and the procedure can be conducted simultaneously on all three major operating systems on the market (Windows, MacOS and Linux). Similarly, Unity allows the developer to compile the game in formats compatible with most platforms and devices: Windows, Mac, Linux, HTML5, PlayStation 4, Xbox One, Nintendo Switch, Android, iOS, etc. Furthermore, the possibility to compile the game in the HTML5 format, so that it can be integrated (and thus become usable) in a web page, removing the installation requirements from the end machine of the user and, thus, simplifying the access to the game, is also of great interest. Similarly, the possibility to compile the game for mobile operating systems, such as Android or iOS, allows the use of tablets and smartphones, both cornerstones of the new digital skills, especially among younger people.

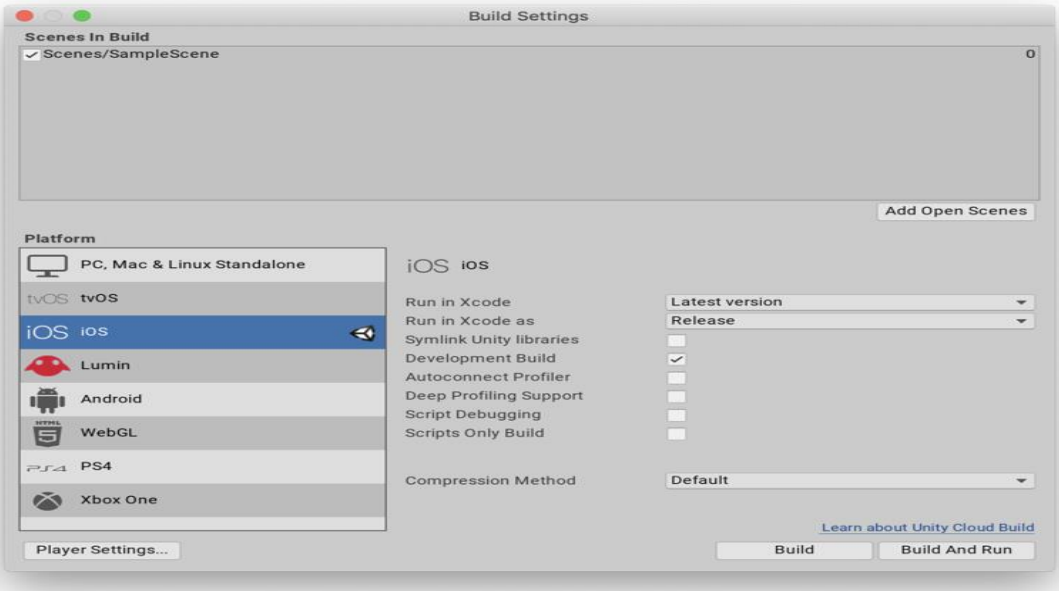

Figure 5: Build settings in the Unity interface

One major advantage of this approach, regarding the development of virtual worlds, is the possibility to allow an "off-line" implementation of the final product, and, in any case, to reduce the need for bandwidth (presented in the disadvantages of OpenSimulator) by the inclusion of the graphical assets (3D Models, Textures and Images) within the resulting game.

Although Unity is a complete and professional game-engine, its UI (User Interface) is fairly simple and accessible even to non-experts. The workflow within the game-engine gives the opportunity to the user to accomplish most of the work using only the tools provided by the platform, in the form of a graphical editor, without writing any code directly. The most common behaviours of the components, meaning objects and characters within the virtual world, exist within the Unity environment, as templates ready to use. In case, however, a need for additional behaviours emerges, they can be implemented using scripts written in C# language. One major advantage of Unity is the possibility to extend the graphic editor through scripts. This allows the addition of custom features to the graphic tools of the editor, in order to provide complete control over the design of a virtual world, or to make the workflow simpler and faster, even for users who lack the ability of programming.

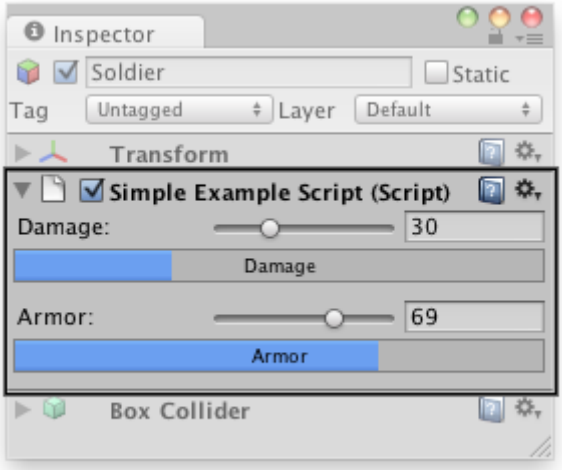

Figure 6: Graphic interface of custom features in the Unity editor

In order to facilitate the work of development teams, Unity Teams has been introduced by Unity. Unity Teams is a collection of features especially designed to make the collaborative experience of game creation easier and faster, allowing immediate saving, sharing and synchronizing the Unity project by all team members. There are, mainly, two services within Unity Teams: *Collaborate* and *Cloud Build*.

Unity Collaborate is a feature that enables sharing and synchronizing the project among all team members. Collaborate allows the users to view and share changes to their project directly from the Unity editor.

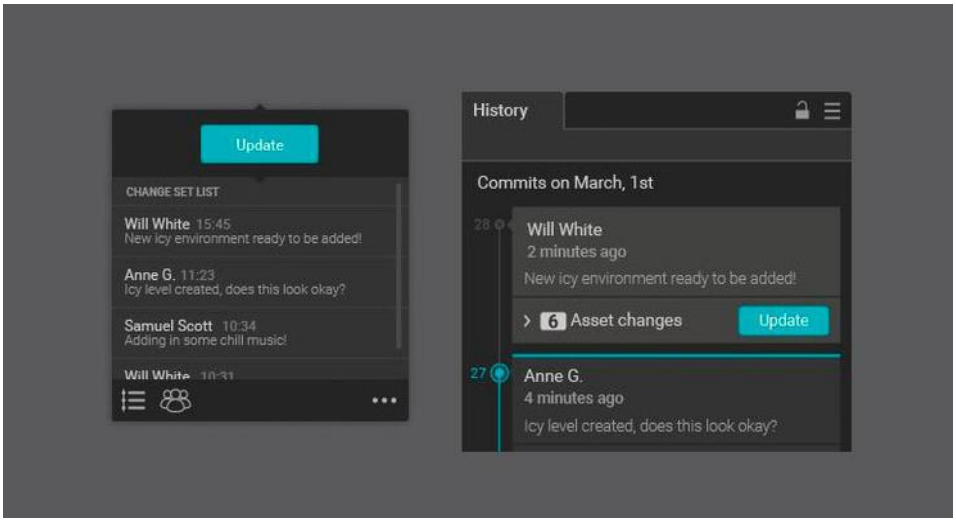

Figure 7: A view of Unity Collaborate

Moreover, Unity Collaborate, allows keeping track of the history of every change made in the project, sharing between the users in real time and retrieving a previous save, in a way similar to a Version Control System – VCS, such as git. Unlike the traditional

versioning tools, all the above actions are possible through a graphical interface which is integrated into the Unity graphical editor, accessible and easy to understand by all the participants to the project. In this way, all the participants can create their own content, without the fear of conflicting with the work done by other team members simultaneously.

Activating Cloud Build offers the possibility to compile, test and distribute the game automatically to the users, because of the dedicated servers in the Unity cloud.

If the project does not reside on the Unity cloud, the Cloud Build can be connected to a repository of a VCS, external to Unity, that contains the current project. Thus, the user-defined pipeline starts automatically whenever a change is saved in the cloud or in the repository, a process that normally includes the execution of automatic tests and the subsequent compilation and, sometimes, distribution of the final product. Additionally, a notification is sent to the whole team, when the execution of the automatic tasks is complete. Due to the build and/or the distribution automation, the above process streamlines the workflow and also, offers the possibility to parallelize the pipelines for every different target platform, such as PC, Mac or Linux.

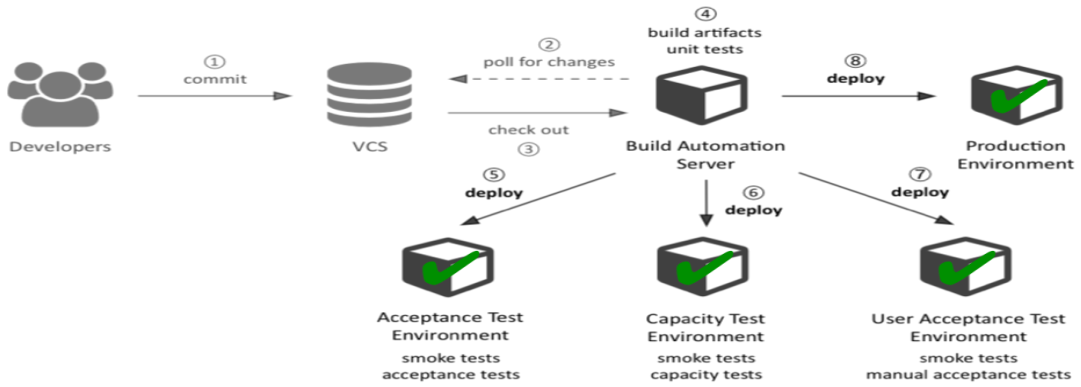

Figure 8: Unity Cloud Build workflow

Sometimes, while defining and implementing graphic assets and "special effects", the user has to deal with the writing of shaders, which are programs that are meant to run on the video card and are written in specific languages, such as GLSL and HLSL. In order to write these programs, the presence of a specialist is often required. The specialist knows those languages and is aware of problems related to the writing of those programs (i.e. the shaders).

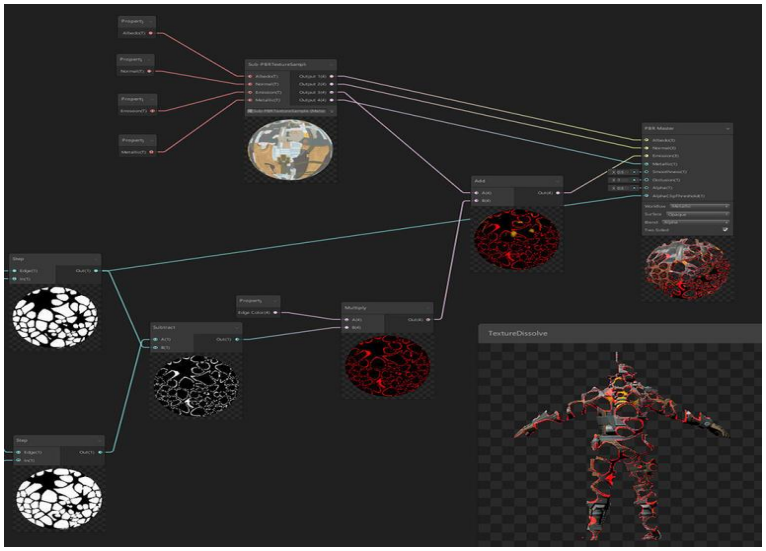

Figure 9: View of Unity Shader Graph

These problems are eliminated by the introduction of two recent tools by Unity; the *Shader Graph* and the *Visual Effect Graph*. These programs are both "nodes-based" graphic editors. The Shader Graph allows the creation of the shaders, eliminating the need of knowledge of GLSL and HLSL languages, while the Visual Effect Graph is used for the creation of special effects. They both present a real-time preview of the final result, characteristic that allows shorter iteration times and that subserves the easier achievement of the final result.

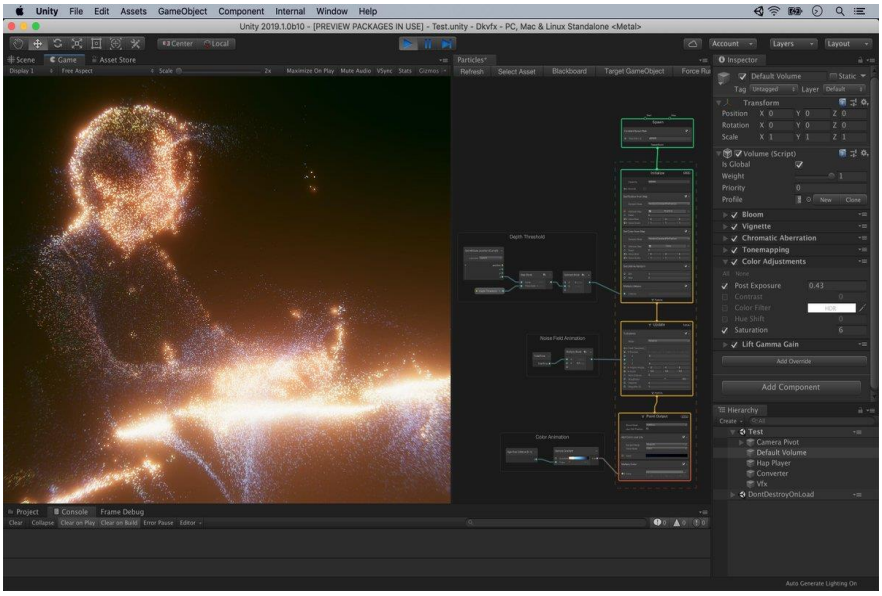

Figure 10: View of Unity Visual Effect Graph

Taking under consideration the limited time and/or the limited professional resources, Unity has another major advantage, offered by the Asset Store. This is a marketplace, where users can buy and sell components, scripts, extensions or graphic assets for Unity. The prices in the marketplace, besides the free assets, are determined by the asset designer, which makes them very competitive. That way, the user has the opportunity, with minimum expenses, to reduce the time and the resources that are necessary to complete the creation of the virtual world. The user can access the Asset Store from the Unity editor and the Unity Technologies web platform. When an asset is purchased, it is added to the collection of the profile used to purchase the asset and it can be added into numerous projects. This offers the chance to create a collection of assets and components, over time, that makes the development simpler and faster.

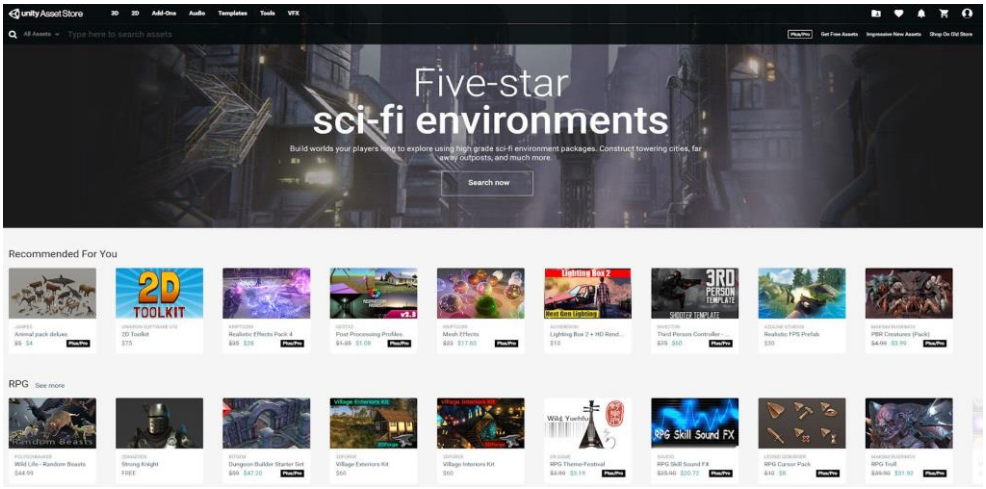

Figure 11: Unity Asset Store home page

In the field of game-engines, Unity is, undeniably, the main player. The characteristics that make Unity the colossus that dominates the industry are the huge community, the "possession" of the 45% of the market share in game development and more than 50% of the share in mobile development, and the amount of educational resources, such as tutorials, guides, training courses, etc.

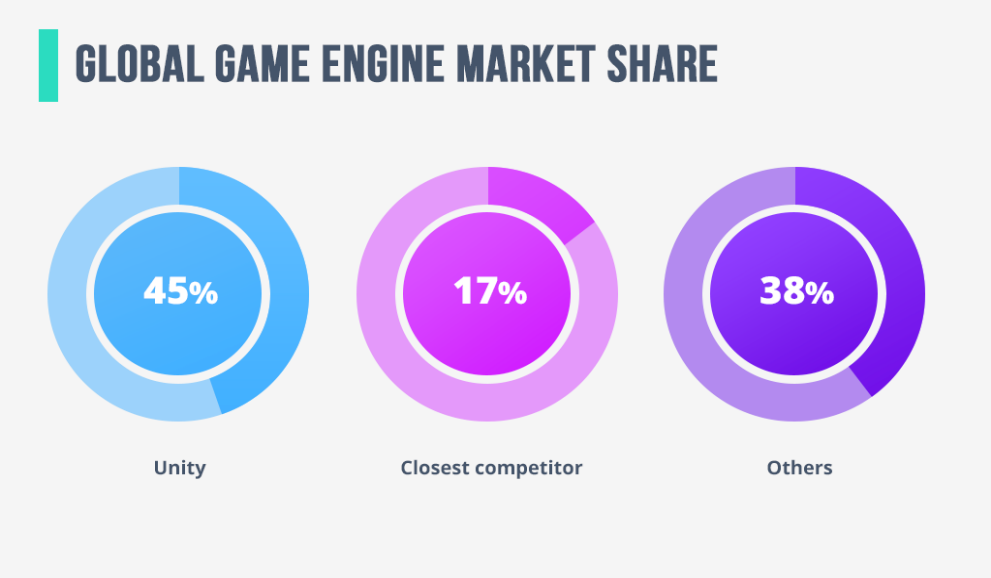

Figure 12: Global game engine market share

Key Characteristics of the free version

- NVIDIA PhysX Physics Engine
- Mechanim: complete animation system for characters and objects
- Professional graphics
- Audio 3D
- Scripting API (supports JavaScript, C# and Boo)
- Multiplayer Networking
- Single-Click game deployment to any platform
- Web-browser integration

2.3 Open-source Platforms

2.3.1 OpenSimulator

OpenSimulator, Opensim- (2007) is an open-source, free, 3D application server for the development of multiuser 3DVWs. The VWs, which can be private or public, can be accessed through various open source clients. OpenSimulator is based on C# and can be easily extended using external modules.

Figure 13: OpenSimulator

OpenSimulator started as an open source server to Linden Lab's Second Life open-source client. Thus, the architecture of OpenSimulator is heavily influenced by that of Second Life, allowing the user to create similar, highly detailed 3D graphical environments for free or at a low cost, provided that the hardware/software and the building/scripting/technical skills are offered for free. Also, the user can fully customize the avatars, resembling those of Second Life.

The communication, in the context of the virtual world, is accomplished by chat and IM. After requested, a voice service with lip sync can be provided by Vivox Inc. An important characteristic of OpenSimulator is Hypergrid, which is a protocol that allows hyperlinking between OpenSim VWs and supports avatar transfers among these worlds seamlessly.

OpenSimulator is very popular among the educational and science communities, because of everything mentioned above and the freedom that is provided to anyone, regarding the owning the owning, the building and the configuring of the VWs.

OpenSimulator is a server - client 3D open source multi-user application, that is used to create virtual worlds that are accessible through an internet connection by the various users of the platform. Creating a world inside OpenSimulator is a relatively simple process, since it represents an already developed and consolidated platform that only requires customization. Once a VW is created, to add content, the user, just needs to log in with an owner of the virtual world account and then they can create 3D models and Textures, or import them from external files. The user is able to create simple solid

geometric figures in OpenSimulator client, which offers an elementary 3D modelling tool. Moreover, once the virtual environment has been created, users can personalize it, in terms of interactions with the elements of the scene. In fact, it is possible to insert scripts that program their behaviour (such as the implementation of a dialog system for non-playing characters, or NPCs) in the virtual elements. This enables the user to implement highly customized experiences within each virtual world in OpenSimulator.

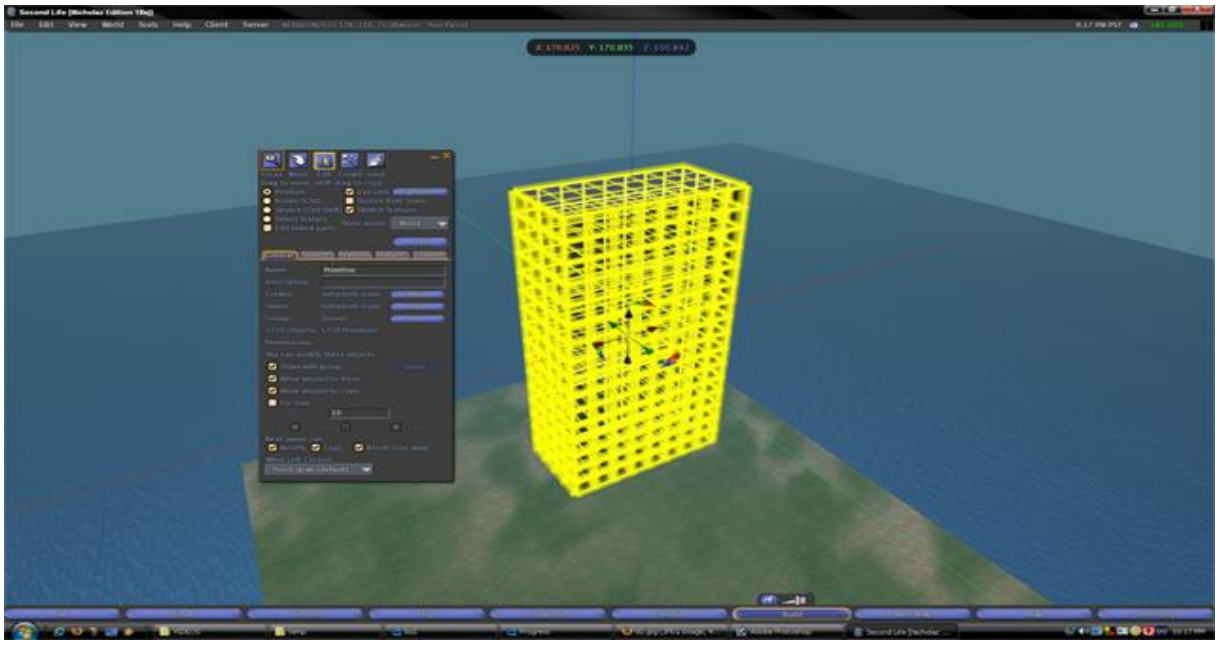

Fig.19 Modelling interface in OpenSimulator

These features renders OpenSimulator an excellent solution in case the virtual world is focused on voice interaction between users (such as a lesson in a virtual class). On the other side, those features are limiting, in case one wants to vary from the proposed game scheme (for example, if one wants to implement particular game mechanics or to use particular hardware, such as sensors, VR viewers, etc.).

In OpenSimulator, the virtual world a user wants to visit, resides on a server, which contains all the resources of the environment, such as 3D models, textures and scripts. So, in order for the OpenSimulator to work properly, it is required that the client application connects and communicates with that server.

The moment that the user enters a virtual world, it is therefore necessary for the client to download all the necessary resources from the server and in the same way, it is essential for the client and the server to remain in communication throughout the user's visit in the virtual world. However, there is a risk of a drop in performance, depending on the complexity of the elements that make up the scene and the quality of the Internet

connection. So, an inadequate bandwidth between server and client, combined with excessive latency of the connection can result in the lack of system response to the user's inputs, because of the loss of the communication protocol packets, or in prolonged loading times of the scene, which is necessary, in order to send and download the 3D models and their textures, as well as in non-fluid movement of the avatar and of the surrounding elements. Specifically, the quality, the characteristics and the number of elements present in the scene (features also related to the download bandwidth available to each client, or player), and the number of clients, or players, that can coexist at the same time within the virtual world, are both determined by the upload bandwidth available to the server. Subsequently, a meticulous evaluation of the bandwidth that is required is needed for a complex virtual world, rich in elements and interactions and designed to accommodate numerous players simultaneously.

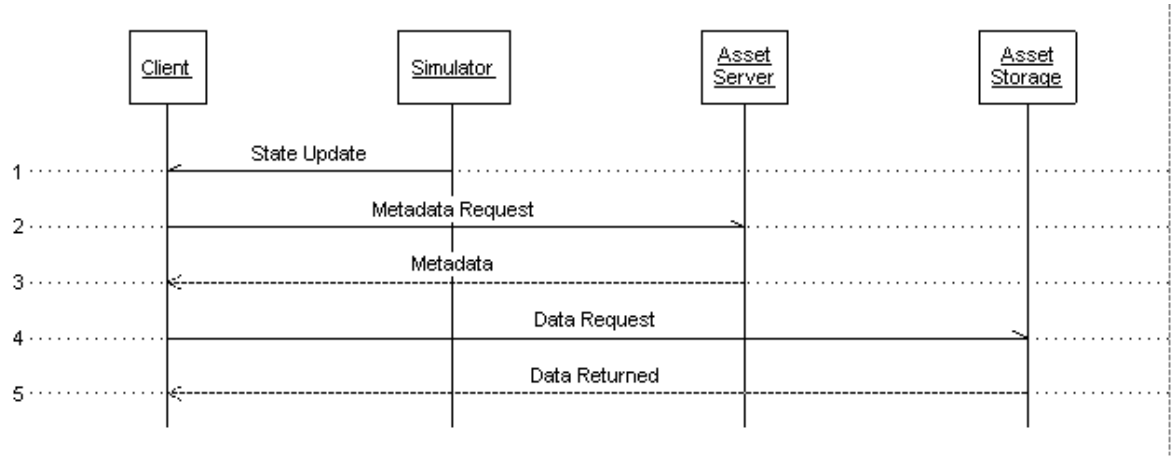

Fig.20 OpenSimulator software architecture

Furthermore, the client needs low levels of latency in order to obtain a rewarding and responsive experience, a feature that is highly dependent on the existing network infrastructure between the server and the client. For example, it could be difficult to locate the server in a region (or even a continent) different from that of the clients.

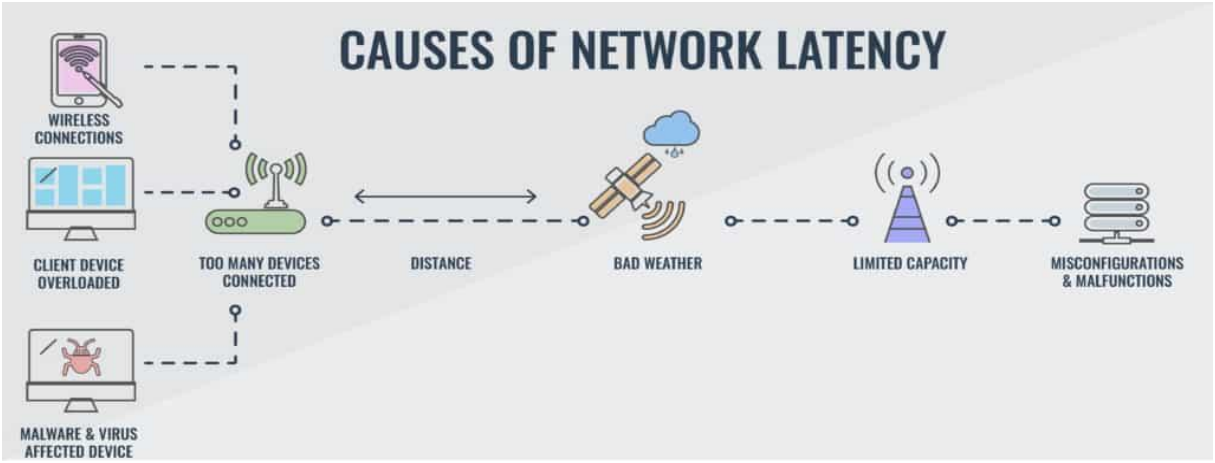

Fig.21 Diagram of network latency causes

An also essential aspect is the synchronization between the server and the client, in order to be ensured that the simulation is consistent and to get feedback in real time, especially when many users participate in the same scene, at the same time.

OpenSimulator is an open source solution, which allows users to set up their own server locally, meaning in the same network or even in the same machine where the client resides. This method helps the user to overcome the network problems listed above, but would require computer skills of an "above average" level, such as network management, database management, etc, to properly configure the server and the connection between the server and the client.

Another requirement of a solution developed on OpenSimulator, is the use of a PC with very precise minimum technical characteristics regarding the CPU and memory, that if are not met, a correct execution of the application is likely impossible.

Moreover, versions of the application compatible with mobile devices, such as smartphones and tablets, do not exist. However, there is a viewer for android devices that only allows the users to view text chat messages and the grid of their inventory. But, it is not possible to access the 3D scenes of virtual worlds through this simplified viewer.

Architecture:

- Server – Client
- OpenSim Server: open-source, multiplatform, free to download
- Client ('viewer'): open-source, multiplatform, free to download

Key Characteristics:

- Physics engine: OpenDynamics (ODE)

- High quality fully customizable avatars
- Built-in 3D building tool
- Image, video, audio file upload (free)
- 3D COLLADA format meshes upload (free)
- Web pages and media on virtual objects
- Terrain modification
- Bots (or NPCs)
- In-world scripting: Linden Scripting Language (LSL), OpenSim Scripting Language (OSSL), C#
- Region and avatar inventory backup and upload
- Text and IM communication in-world.
- Voice not built-in: through external free modules: Vivox, Freeswitch, Mumble
- Hypergrid protocol
- Access and building rights restriction
- Groups and full user profiles
- Head Up Displays (HUDs)

2.3.2 OpenWonderland

OpenWonderland, OW- is an open source Java toolkit for creating 3DVWs. OW is in early stages of development. So, the graphics are rather simplistic but other features of the platform are comprehensive. The toolkit allows the creation of modules that can extend the functionality of the server or the client. Moreover, customized, special-purpose VWs can be created.

Figure 14: OpenWonderland

Examples of the external modules that have been created by different developers and can be found in Module Warehouse are: Authentication system, webcam viewer, writable text/HTML poster, collaborative text editor, and more.

A feature that distinguishes Open Wonderland is the easy embedding of existing content. There is an enormous list of document types that can be added to the virtual world by simply "dragging and dropping". Moreover, the user can import any content within the Google 3D Warehouse. Open Wonderland does not offer in-world 3D building, but 3D objects can be designed using platforms like Maya, Google SketchUp, Blender, etc. and then be imported into the world.

The users can communicate with high-fidelity, immersive audio, share live desktop applications and collaborate in an education or business context (simulations, meeting rooms, mixed-reality worlds, etc.), inside the Open Wonderland VWs.

Architecture:

- Server – Client, open-source, both provided free

Key Characteristics:

- In-world embed scripts in 3D models
- Scripting in Javascript, PHP, Groovy, JRuby, Java, Jython
- Drag-and-drop content
- In-world run Open Office, Firefox, NetBeans, etc.
- Shared application framework (whiteboard, PDF viewer, sticky notes, etc.)
- Screen sharing

- Webcam viewer, video player, audio recorder
- Integrated telephone
- High quality immersive audio
- Group and private text chat, private voice chat
- Portals creation to locations on same server or on different servers ("teleportation")
- LDAP plug-in for connecting to existing LDAP authentication systems

2.3.3 Open Cobalt

Open Cobalt, OC - is an open source VW browser and toolkit for creating VWs. Instead of the server - client schema, OC uses peer-to-peer architecture, which enables users to access OCVWs on LANs, intranets, or across the Internet, without the requirement of accessing remote servers. In this manner, anyone can host an OCVW for free.

Figure 15: Open Cobalt

The interactions within the VWs are conducted through Open Cobalt's peer-to-peer technology. This technology makes OC differ from other commercial multi-user VW platforms where in-world interactions are done through central servers, such as Second Life. The main advantage is that the set-up of VWs and the interaction between users of OC can be performed without any hosting fees, licensing or virtual land lease costs.

OC allows users to form large distributed networks of interconnected collaboration spaces, by hyperlinking their VWs and. Additionally, it allows the set-up of 3D virtual workspaces, public or private, that feature integrated web browsing, voice chat, text chat, and access to remote desktop applications and services.

However, OC does not support in-world 3D content creation tools. It provides the infrastructure for world creation, navigation and collaboration and regarding the virtual content of the world, it supports content created in open source applications such as Sketchup or Blender.

Architecture:

- No Server: Peer-to-peer technology instead
- Client: Open Cobalt Browser (free and multiplatform)

Key Characteristics:

- Avatar selection from custom avatars
- "Squeak" for in-world and system scripting
- Basic end-user content creation and editing
- 3D objects, mesh, texture, media (audio, videos) import
- Navigable 3D hyperlinking between virtual worlds
- Enterprise directory access (LDAP)
- In-world annotations (text and audio)
- Text chat, in-world voice chat (through VoIP)
- Collaborative document sharing/editing
- Save/restore virtual worlds
- In-world web browsing (via VNC)
- Access to remote applications (via VNC)

3 VW PLATFORM FOR THE VR-WAMA PROJECT

Regarding the VR-WAMA project, there is a need for a 3DVW environment and platform that is technically mature, as well as immersive, providing a sense of presence to the users. Besides the quality of graphics, that must be high, immersion and sense of presence also refer to avatar personalization and easy, straightforward in-world interaction.

Moreover, VWs used for educational purposes, programming capabilities might need to be included, in order to provide interactive objects and special functionality, such as bots (NPCs) that will be able to support the scenario requirements, as well as mechanisms for tracking the user's performance and providing feedback and recommendations.

Several VW platform selection criteria have been identified, that are in accordance with the VR-WAMA project goals:

1. Cost-free and open-source
2. Allow the development of fully customizable and multiuser VWs, in order to simulate various scenarios
3. Good system stability
4. Straightforward server configuration and parameterization, in order to allow full control over the VW and the usage rights at will
5. Self-hosting possibility
6. Reasonable hardware and bandwidth requirements
7. User-friendly, free downloadable, multi-platform client software
8. High-quality 3D graphics and human-like fully customizable avatars, in order to support immersion and sense of presence
9. The platform must be popular among educational projects, while a large, active and supportive community of developers should be present

Based on the first criterion, the chosen platform must be open-source and free. Taking under consideration the features and the functionality of the open-source platforms presented in Section 3 of this report, OpenSimulator seems to be the most suited option for VR-WAMA. OpenSim is more mature and fulfils all the aforementioned requirements. Another aspect that suggests that OpenSim is the ideal platform for educational

institutions and enterprises that need to have full control and maximum flexibility on their 3D simulations, is its compatibility to SL, the most popular 3D virtual world platform for educators worldwide. Also, OS's open and modular design, makes the OS platform ideal, while the VWs offer graphics of similar quality to Second Life, similar functionality and similar building possibilities as Second Life but with significantly reduced cost - or for free.

Moreover, another major advantage of the OpenSim, especially for developers not so experienced in this field, is the existence of two very large and active communities, one consisting of educators and one of developers. Although OW and OC platforms present interesting features, they are still in earlier stages of development and their evolution is slower, in comparison to OS, while they are supported by smaller community of developers (especially the OC platform).

4 THE OPENSIMULATOR PLATFORM

4.1 Introduction

OpenSimulator is an open source 3D application server that can be used to create a Virtual Environment and is accessible through a variety of clients. Virtual environments can be simulated in a similar way to Second Life, supporting the core of SL's protocols and functionality.

The community of people and groups which contribute to the development of OpenSimulator and to the creation of open content, is very active. In the field of education, several universities and other educational institutes have integrated OpenSimulator as part of their courses and have successfully used it for innovative research on learning methodologies.

In this chapter, many of the features offered by OpenSimulator will be presented and described.

4.2 Architecture

OpenSimulator is a server application that runs on one or multiple machines (when used in grid mode) and serves client applications, or 3D Viewers, through HTTP messages and the Second Life protocols.

The OpenSimulator server can be used in two modes. In the first mode, the standalone mode, the whole simulation is run as a single process, in order to run only in a single machine. This makes the configuration much easier, but it does not allow the connection of the hosted virtual world with other Virtual Worlds online. In the alternative mode, the grid mode, various aspects of the simulation can be separated and run as different processes across different machines. This allows multiple Virtual Worlds running in different servers across the internet, to share common users and assets data, while giving users the ability to easily teleport from one to another, who can maintain their inventory items. Thus, a hypergrid of connected worlds is formed, which is effectively supporting the emergence of a Web of virtual worlds.

OpenSimulator is written in C# and is designed in a way that allows to be extended through modules. Also, it uses a database to store almost all content of the 3D Worlds and it can be configured to use any of the popular database software (mysql, postgres e.t.c).

The communication between server and client is achieved through a command line program running on the server. This program is responsible for the communication with the clients and it ends the necessary information that will be displayed on the client's machine. Consequently, the server machine does not require 3D graphics card with high demands. On the other hand, client machines require a machine with a good GPU, since the rendering will take place in them. Also, it is important, for the server running the simulation, to have a sufficient amount of memory, particularly if many users (client computers) connect simultaneously.

4.3 Installation

To Run OpenSim, the user will need:

For Windows: .NET Framework 3.5

For Linux : At least Mono 2.4.3

OpenSim only requires extraction of the archive file and not installation. There are several important configuration files with parameters (such as credentials for connecting to the SQL server etc) that the administrator can configure. In order for a simulation to initialize and run, the administrator runs the OpenSimulator executable and then has access to a console that displays logging messages about the communication with clients and that allows using a set of commands, by which, the administrator, can create users and regions, import terrain height maps or folders of items in inventories and many other tasks regarding the administration.

4.3.1 Windows installation

The only requirement is the download and extraction of an archive, which contains the files of the platform. In Windows, the user should run "OpenSim.exe" (.NET Framework 3.5 is required). The first time the application is run, it will guide the user through some steps, in order to initialize and configure the 3D World. Among others, the creation of the first region and of the owner account are requested.

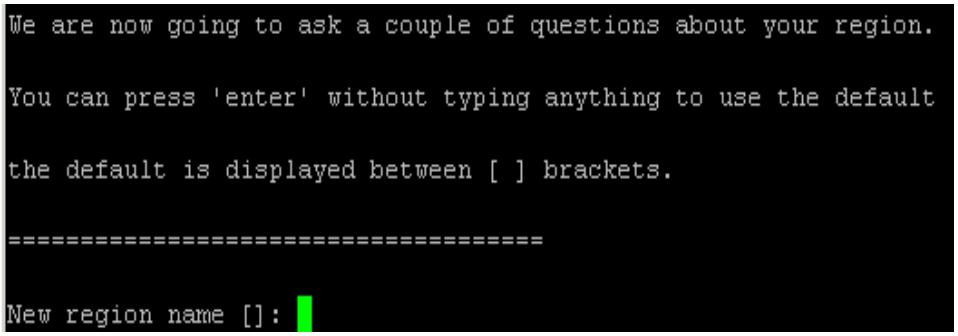

Figure 16: Region Creation

Depending on the user's preferences or needs, the installation of some additional software might be needed, in order to enable some features:

- **Freeswitch** server for Voice communication. This will allow the use of microphones by users connected in the 3D World, in order to communicate through speech with other avatars.
- **Apache server with PHP** is required for some modules to work (such as offline messages or groups), but is not necessary to run the simulation.
- **MySQL Server** or other databases can be configured to be used instead of the default, which is already embedded in OpenSim (MariaDB). This is recommended in order to achieve improved performance.

4.3.2 Linux installation

The steps for installing OpenSim are almost the same as those mentioned above. The only difference is that the user needs to have "mono" (at least version 2.4.3) in order to be able to run the OpenSim application. If a remote access to OpenSim's console is desired, a helpful idea is to also use "screen", that allows the disconnecting and re-connecting to shell sessions from multiple locations.

4.4 3D Viewers and User Accounts

The visitors can access the Virtual World through a special type of software, called a 3D Viewer. There are various alternative 3D Viewer options, such as Kokua, Firestorm, Imprudence and Singularity, which share the same functionality, although they have some differences in the user interface.

In order to access the 3D world, users need to create a "user account" and this is usually done via a web site interface, where they submit a name for their avatar (First and Last name) and a password. There are different alternative user management interfaces, such as Wifi, based on different technologies. If the user does not desire to use an interface, then the server's administrator would have to create accounts for interested users, that want to visit the 3D World, manually.

The user needs to know the server's address and port on which the simulation is running, to establish a connection to a specific Virtual World running on a distant server (similarly to the adders of a web page) and provide the credentials for the avatar that was created as described in the previous step.

4.4.1 Setting up user registration system

At first, there is no user registration system in OpenSim. The administrator will have to manually create user accounts, using the OpenSim console. However, the interested users have the ability to create their own accounts, using extensions that are available and that allow setting up a web-based registration system. These accounts can be activated automatically, or require approval by the administrator first.

Wifi (http://opensimulator.org/wiki/Wifi) is one of the popular user registration options and it is relatively easy to install. Some additional files need to be downloaded and extracted and then, some parameters in specific OpenSim configuration files need to be edited. After that, the creation of some default avatars (e.g. Male and Female) follows, so users can choose their initial appearance when they create an account.

4.4.2 Overcoming barriers: firewalls, hardware and software limitations users may experience

In case the Virtual World is not accessible, one of the problems that commonly rises is that a firewall in the server may block incoming connections of clients, which is, actually, the default behaviour in most operating systems. So, it needs to be ensured that the OpenSim application is not blocked by any firewall, and the ports required by OpenSim are available. By default, OpenSim uses port 9000, however, the opening of additional ports may be needed (a new one should be specified for each Region created). If additional features are used, the same process may be needed to be followed with other applications, such as Apache or FreeSwitch Server.

The most popular 3D Viewer applications provide versions for Windows, Linux and MacOS. There is also an implementation of a 3D Viewer for Android devices available (Lumiya viewer), but with some limitations. Unfortunately, there are some hardware requirements for these applications to run, mainly regarding the GPU card.
So, sometimes, the installation of the latest version of the graphics card drivers may be necessary for the 3D Viewer to run.

4.5 Regions

Regions are individual areas inside the 3D world, which can contain multiple regions. Each region has a location in the map of the world, and can have other regions in the four adjacent, neighbouring areas. By default, the regions are independent of each other and users teleport from one region to the other, even when they move to neighbouring ones. However, a recent feature, allows many regions to be combined, so that there is no border transition between regions and is called "megaregions". Thus, the users can move from one region to the next, without having to teleport, avoiding teleportation delays.

4.6 Terrains

The ground surface of the regions is called terrain and it is configured through a heightmap that indicates the distance of each individual position from the lowest plane. A specific height can be set as the sea level, so anything below will be filled with water. Furthermore, custom, separate textures can be used for different height ranges, as desired by the creator of the 3D World, allowing, for example, rocky textures for higher areas (e.g. mountains) and grass textures for lower areas. The heightmap can be set from an avatar (with ownership rights), manually, inside the Virtual World, using a variety of tools provided (flatten, raise, smooth etc). Also, there are automated tools, such as L3DT, that are able to generate a heightmaps from images or other sources and then be imported in regions of a virtual world. Moreover, in order to maintain a smooth continuity between different regions, it is possible to have a larger height map be spread over multiple regions. Last but not least, one can use actual real height data, from

relevant datasets and generate heightmaps that accurately correspond to areas in the real, physical world (e.g. a specific island in the world), in an easy manner, taking advantage of various tools available online.

4.7 3D Objects

Similar to other virtual worlds or 3D modelling software, 3D objects can be added in a region by creating and manipulating simple objects called "prims". At first, the user creates one of the basic prims (cube, cylinder etc) by dragging and dropping them, from a menu in the client program, in a position, and then can use corresponding controls to scale, move or rotate them.

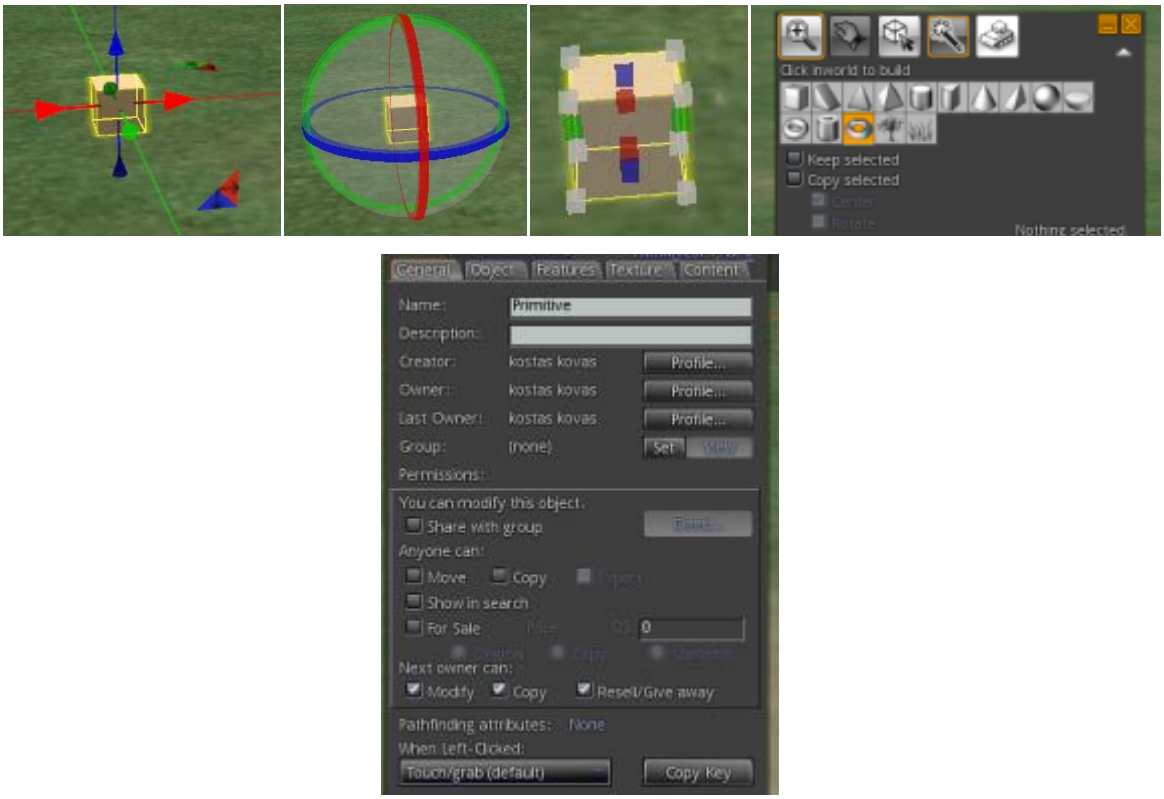

Figure 17: 3D object examples

Each prim has a menu with various options/tabs to modify its parameters:

- General: Contains general information, such as the name and description of the object, the Owner and the permissions of others.
- Object: The location (x-y-z location) of the object inside the world, its rotation, size and type. An object can be physical, so it follows physical laws as defined by the physics engine, if used.

- Features: These options allow configuring the lighting cast on the object or a Flexible Path that allows it to behave like complex, dynamic objects (like cloth, hair etc).
- Texture: This tab offers options that allow to set specific texture to each distinct plane of an object.
- Content: Every prim has its own inventory and so can store other objects, but, most importantly, script files that contain code (using LSL syntax) that defines its behaviour.

The parameters for the modification of the shape of an object are extensive and allow creating intricate, complex objects. A very important aspect is also the ability to connect multiple prims into link sets. Doing so, the individual objects become parts of a single object. This also facilitates the interaction between linked prims. For example, it is possible to link one object to another and add scripts so that the first object rotates around the second.

A user can learn to create complex, detailed objects, by joining simple prims, but there is also the alternative to import 3D models designed in more advanced modelling software, such as Blender, Modo, SketchUp etc. This is possible through the widely used Collada format. Also, one can find many free (or low cost) existing 3D models in Collada format, that can be imported directly in OpenSim, provided by online communities.

4.8 Avatars

When users enter the virtual world, they control an Avatar character, which represents them. The avatar is an aspect of high importance, in order to achieve a high degree of immersion. Avatars are customizable and, by modifying body parts (Shape, Skin, Hair and Eyes), clothing or using various items as attachments, they can have the desired appearance.

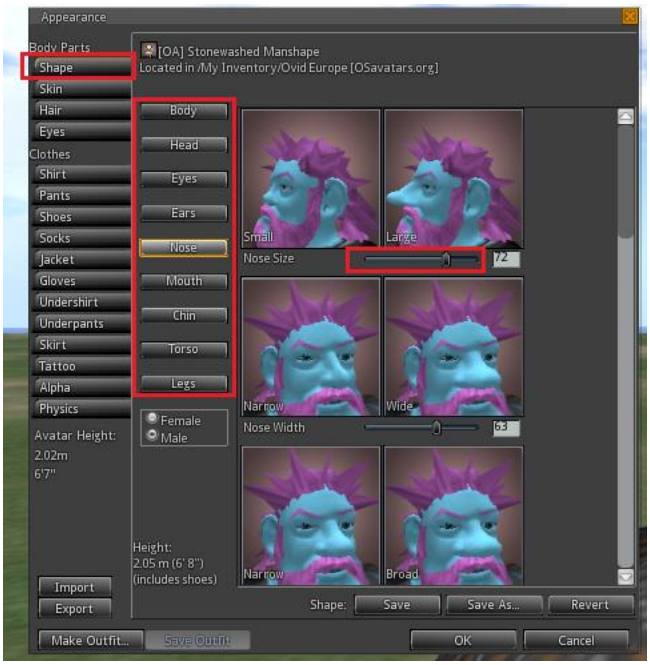

Figure 18: using sliders to adjust the shape of an avatar

Figure 19: a custom avatar using skin textures and attachments

Instead of using the default shapes, it is even possible to create a custom 3D model and rig (the skeleton of the character) for the Avatar, using 3D modelling software. Also, custom textures can be used for skin, hair and eyes, resulting in very detailed and realistic characters.

A set of different clothing types are available and can be adjusted, using menu sliders but one can also create custom, complex ones, using the attachment slots. "Attachments" are 3D objects (single prims or link sets) that can be "attached" and then adjusted to specific slots of the character (e.g. left hand to have the avatar hold the item, head to wear a hat or helmet, chin to attach a beard). It is, also, possible to create attachments with realistic

behaviour, such as hair flow when the character moves, or according to the direction of the wind, by using the "flexible path" parameters of an object.

By adding custom animation files in the popular motion capture format "bvh", that can be found online, in many datasets that contain motion capture data, one can override the animation of the character (e.g walking or flying around). These files may have been made manually or automatically, through motion capture techniques, recording the animation of real people and they can be used directly or with minor adjustments. Also, by using software, like QAvimator, it is not difficult to create a custom animation. QAvimator displays an Avatar and rig of an OpenSim character and allows setting the position of each body part, for every time point in the timeline. Animations can be triggered inside the world manually by the user, by selecting and "playing" an animation file in his inventory, or as a result of an action (e.g touching an object) or a script.

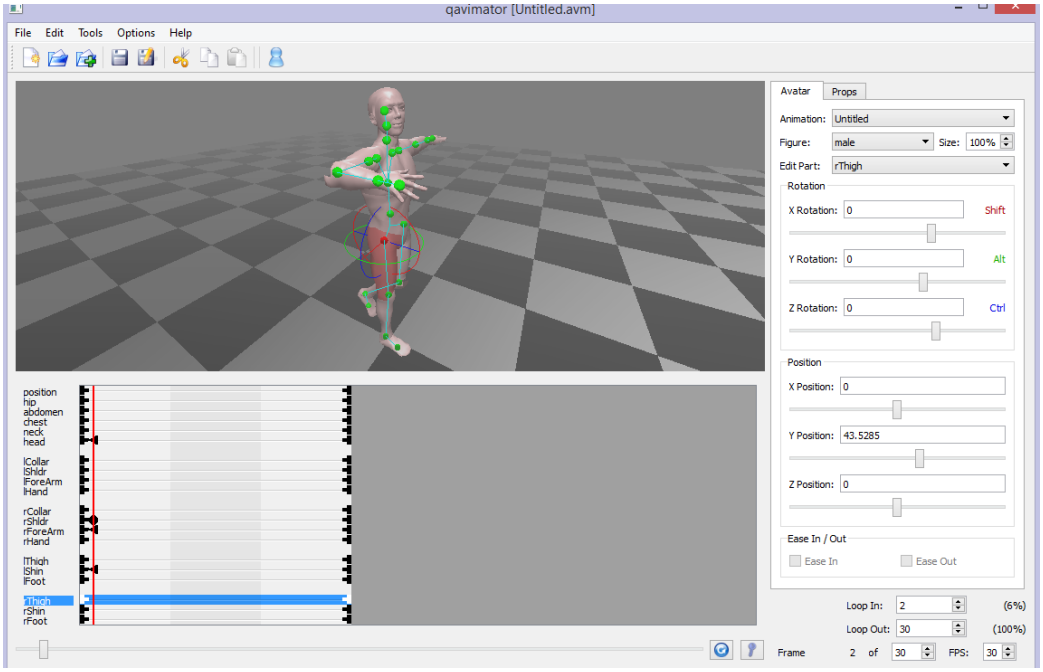

Figure 20: creating a custom animation in QAvimator

Controlling the character and the camera view is a fairly easy procedure and usually is done using the same keys used in other Virtual Worlds or popular 3D Video Games. Avatars can walk, run, crawl and even fly around, unless there is a restriction by the region's administrator.

4.9 Textures

The 2D images that are applied on the planes of 3D objects are called textures. In order to achieve detailed and realistic virtual worlds, one should create quality textures for the virtual objects. This is a demanding procedure, especially for complex objects, referred to as "UV mapping". However, when working on basic and simpler prims, with limited plane, UV mapping becomes easier.

OpenSimulator also allows the application of a mask over the textures, to easily modify the colour, add a glowing effect on it, or even setting a transparency value, giving it an opaque view.

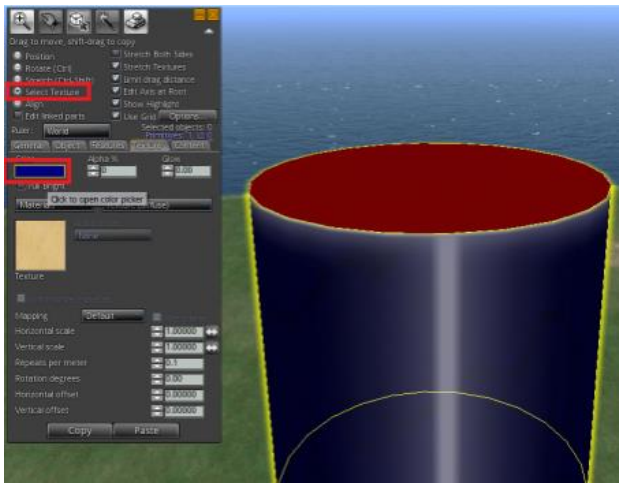

Figure 21: colouring textures of a cylinder prim

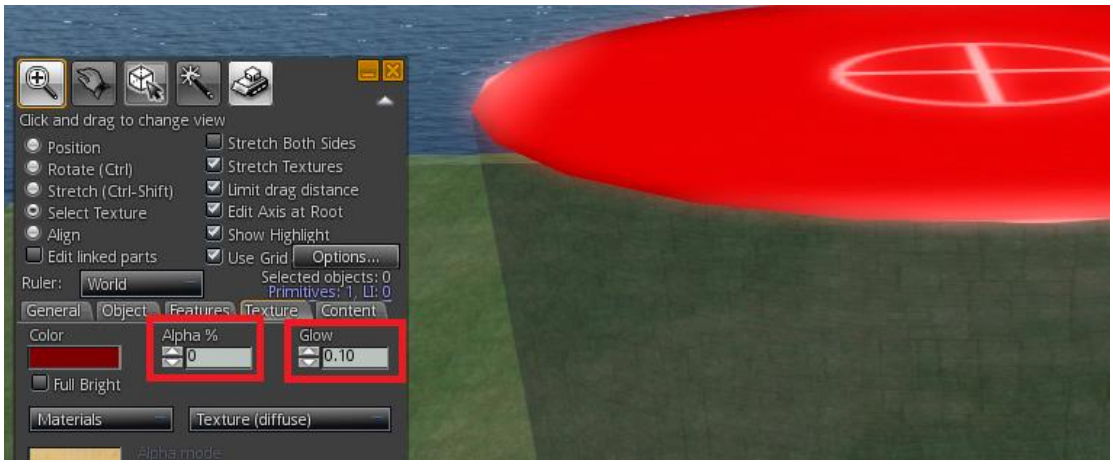

Figure 22: modifying transparency and adding glowing effect

Also, image formats with transparency capabilities, such as PNG, can be used to make simple 2D objects appear as 3D when viewed from specific angles.

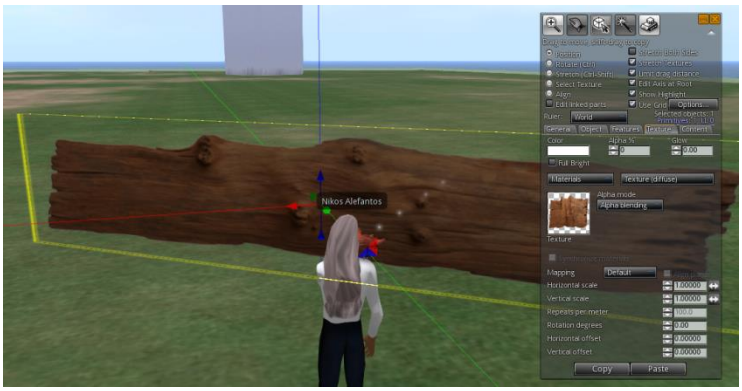

Figure 23: using an image with transparency to create a 2D tree trunk.

Finally, a recent feature is MOAP (media on a prim), that allows projecting a web site on the surface of a prim. This offers an important aspect to trainers, since it allows them to project existing learning material (e.g. text or presentations) inside the Virtual World, without the need of creating 3D objects.

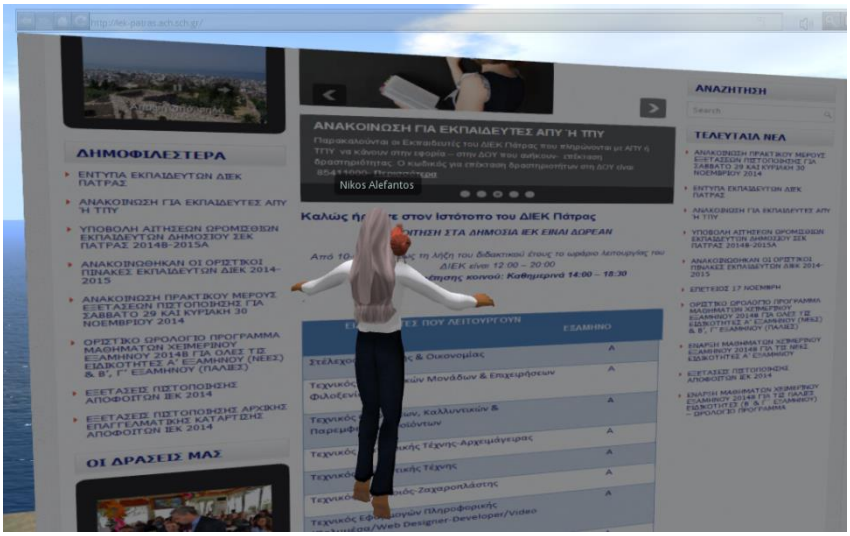

Figure 24: projecting a website inside the 3D World

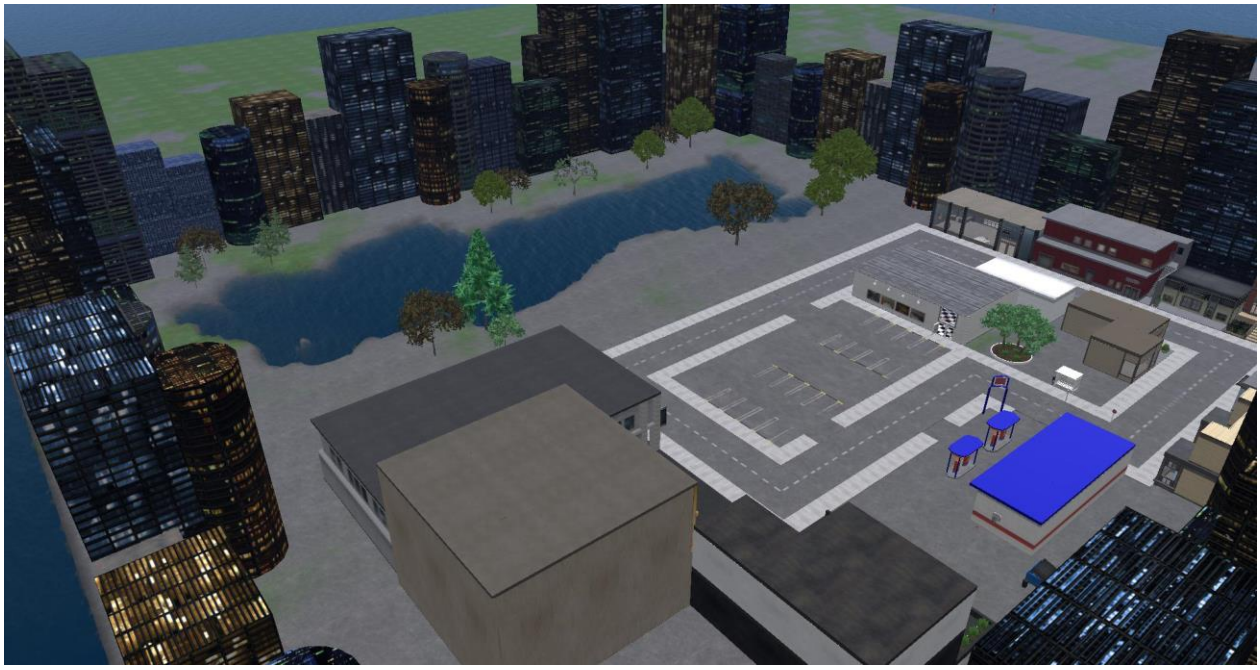

Figure 25: Part of the VR-WAMA City

4.10 Physics Engine

Physics behaviour, for specific objects, can be simulated using a number of alternative modules in OpenSimulator. Setting a particular object as "physical" will cause it to fall towards the ground or respond to collisions with other objects or avatars.

4.11 Communication

The interaction with other avatars (representing other users) can be accomplished through text messages, which can be public (displayed by all nearby avatars) or private (visible to specific only users), voice, if a microphone is connected, or even gestures and animations of the avatar.

Voice communication can be enabled by running and configuring appropriate software like FreeSWITCH, since is not present by default but. Also, storing offline messages that will be displayed to users when they log in the world can become possible with the assist of a module that can also be added (the default setting requires both users to be online to communicate with messages).

Figure 26: an avatar's inventory

4.12 Basic Interactivity

Avatars can interact with various objects in the 3D world by touching (clicking) an object or by other events or actions, that can trigger a behaviour that has been scripted in the objects. For example, clicking on a chair object may cause the avatar to sit on it.
Every avatar owns an Inventory, a file directory in which various types of files are stored, such as 3D objects/prims, sounds, animations, scripts, body parts, clothes etc.

4.13 Scripting (LSL)

To describe the behaviour of an object, there are used scripts, which are files written in the LSL programming language, developed by Linden Labs for Second Life. As mentioned earlier, each prim disposes an inventory, which allows the prim to contain one or more script files. Inside a script, different states for the object can be defined and described. For each state, in order for a script to be executed, multiple event listeners can be added and wait for a specific event. For example, a "touch" listener will be triggered whenever an avatar "touches" (clicks) an object.
Some common event listeners are the following:

- wait for an avatar to click (touch) the object
- wait for an avatar to collide with the object
- wait for an avatar to approach the object (specific radius)
- wait for a message to be sent to a specific chat channel

- wait for specific time interval to pass (e.g. every 10 seconds)
- wait for an object to be attached to an avatar
- wait for the state of the simulation server to change (e.g. when the server starts)
- wait for the object to be rezzed (inserted in the 3D World from a user's inventory)

An object can be programmed to perform various actions by using corresponding commands in the body of the listeners mentioned above, some of which are:

- rotate around a specific axis or relevant to another object
- change its size
- change state (so other listeners will be used)
- move to a specific position or move specific units towards a direction
- change the transparency of a texture
- make a plane of the object glow
- change or move the texture applied in one or all of the object's planes (this can be used to simulate running water or similar effects)
- play a sound file (up to 10 seconds) stored in its inventory
- write a text message in a specific chat channel
- show a particular message as a label over the object
- create a dialogue menu for a specific avatar to provide an answer to a question
- give an object to a specific user (added in his inventory if accepted)
- make an avatar teleport to another location inside the Virtual World (or other worlds if running is grid mode)
- change the speed of the avatar
- wait (sleep) a specific time interval before the next command.

As an example, a lamp object is initially in a state called “off”, waiting for the event of some user clicking/touching it. When a user clicks it, it executes an action that makes the lamp turn on (emits light) and then change to another state called “on”. While being in state “on”, the object (lamp) waits for a user to click it, in order to turn off and return to state “off”.

One of the most useful (if not the most useful, albeit difficult to apply) feature of the scripting language is the ability of objects to send messages to specified chat channels,

while being able to listen to messages in other channels. This allows objects across the simulation to communicate with one another.

4.14 Non Playable Characters

Non-Playable Characters (NPCs – i.e. avatars that are not controlled by actual users) can be generated an controlled by using a specific set of scripts

Some of the actions available to control the NPCs are:

- NPC character is created at target position
- NPC rotates to face towards a direction.
- NPC walks (or runs, flies or uses any other animation) to a target position.
- NPC randomly wanders around a point.
- NPC follows the user-avatar.
- NPC does some animation (wave, point, dance, attack etc).
- NPC says a message (text) in nearby chat that appears on screen.

The above actions can be used along with the scripts described earlier to program a detailed behaviour of the NPCs, based on events triggered by real avatars or the environment.

For example, it is possible to create an NPC guide, that senses a real user approaching, and offer to show the avatar around the Virtual World, while talking (using sound files) or displaying text messages.

NPCs can also act as responsive dialogue agents, that ask questions and respond to the users' answers (through dialogue menus or by writing messages in specific chat channels) or as simple, static bots, that just give specific information.

4.15 Integration with LMS

A very important capability of OpenSimulator and the LSL scripting language is the existence of commands that allow the sending and receiving HTTP messages, communicating, this way, with other servers or applications, as well as querying and administrating a database. This makes it possible for external applications, such as custom web sites or even Content and Learning Management Systems (LMS), fetch and

display data from the Virtual World or send and manipulate data inside the Virtual World.

Integrations with LMS environments have been developed, taking advantage of the capabilities mentioned above, with most notable among them being SLOODLE, an integration of OpenSim/SecondLife with a popular LMS platform, Moodle, offered as a free and open source project. SLOODLE provides a range of tools for supporting learning and teaching into the immersive virtual world. These tools are fully integrated with a tried and tested web-based learning management system used by hundreds of thousands of educators and students in a global scale.

Some of the features offered by SLOODLE are:

- Web Intercom: allows students participating in chats in OpenSim using the accessible Moodle chatroom, with all discussions archived securely in the Moodle database.
- Presenter: allows uploading slides in Moodle and having them available inside the virtual world.
- Assignment Drop-Box: allows managing in world assignments through the moodle interface.
- Quiz chair: allows avatars taking quizzes inside the world and managing the results by integrating the Moodle gradebook.
- Choice: allows capturing feedback from users in the 3D World.
- Scoreboard: can be configured to track progress of the students.
- Vendor: can be used to distribute inventory items.
- Rezzer: can store and recreate scenes of 3D objects and activates.
- Reg-enrol booth: allows Moodle Users to be associated with corresponding avatar characters in the world.

5 REFERENCES

1. Bartle, Richard (2003). Designing Virtual Worlds. New Riders. ISBN 0-13-101816-7.
2. Daden Limited. (2010). Virtual Worlds for Education and Training. White Paper. Retrieved February 22, 2011 from http://www.daden.co.uk/media/white-papers.
3. de Freitas, S. (2008). Serious Virtual Worlds: A scoping study. Joint Information Systems Committee.
4. Bell, M.W. (2008). Toward a Definition of 'Virtual Worlds'. Journal of Virtual Worlds Research, 1(1), 2-5.
5. Mikropoulos, T. A. &Natsis, A. (2010). Educational Virtual Environments: A Ten Year Review of Empirical Research (1999 – 2009). Computers & Education, 56(3), 769-780.
6. Nowak, K., &Biocca, F. (2001). The influence of Virtual Bodies and Agency on Co-presence, Social Presence and Physical presence. In Proceedings of 2001 of the 4th Annual International Workshop PRESENCE 2001. Temple University, Philadelphia, PA, USA.
7. Richter, J., Anderson-Inman, L., & Frisbee, M. (2007). Critical engagement of teachers in Second Life: Progress in the SaLamander Project. In Proceedings of 2007 Second Life Education Workshop (pp. 19-26). Chicago, USA. Retrieved October 20, 2012 from http://www.garito.it/prog/psico08/testi-def/slccedu07proceedings.pdf.
8. Mantovani, F., &Castelnuovo, G. (2003). Sense of Presence in Virtual Training: Enhancing Skills Acquisition and Transfer of Knowledge through Learning Experience in Virtual Environments. In G. Riva, F. Davide, & W.A IJsselsteijn (Eds), Being There: Concepts, effects and measurement of user presence in synthetic environments (pp. 167-181). Amsterdam, The Netherlands: Ios Press.