OpenSimulator based Multi-User Virtual World: A Framework for the Creation of Distant and Virtual Practical Activities

MOURDI Youssef, SADGAL Mohamed, BERRADA FATHI Wafaa, EL KABTANE Hamada
Faculty Semlalia
University Cadi Ayyad
Marrakesh, Morocco

***Abstract*—The exponential growth of technology has contributed to the positive revolution of distance learning. E-learning is becoming increasingly used in the transfer of knowledge where instructors can model and script their courses in several formats such as files, videos and quizzes. In order to complete their courses, practical activities are very important. Several instructors have joined Multi-User Virtual World (MUVW) communities such as SecondeLife, as they offer a degree of interrelated realism and interaction between users. The modeling and scenarization of practical activities in the MUVWs remains a very difficult task considering the technologies used by these MUVWs and the necessary prerequisites. In this paper, we propose a framework for the OpenSimulator MUVWs that can simplify the scenarization of practical activities using the OpenSpace3D software and without requiring designers to have expertise in programming or coding.**

Keywords*—*E-Learning; multi-user virtual world; practical activities; OpenSimulator; virtual reality; virtual laboratories

I. INTRODUCTION

In the current circumstances, the use of Information and Communication Technologies (ICT) in both 'distance' and face to face training and particularly in the form of virtual learning environment such as learning management systems (LMS), has become a necessity for universities, high schools and even training companies, consequently they tend to virtualize their education and training system [1], this is due to all the features offered by these virtual environments including document sharing, collaboration, discussion, and communication between learners and their teachers [2].

It should be noted that providing a quality educational experience for all courses type is not easy, some of them like engineering courses require practical activities (PA) and therefore the laboratory concept is very important in the learning process [3][4].

In fact, PAs enable learners to develop useful skills for their professional career, to learn how to handle and use things accurately, to discover new things, and even to test the veracity of an idea. For teachers, practical activities are the best way to build knowledge, to understand facts and scientific theories and to stimulate learner's motivation. This can be done by performing concrete experiences, developing psychomotor skills and/or illustrating scientific approaches to solve problems.

A huge quantity of solutions has been attempted in order to propose approaches for teachers to integrate PAs in distance learning. The first was the remote-control laboratory [5][6][7][8], then the use of videos and finally virtual laboratories [9]. However, the latest proposal is very superficial, and very limited.

At this level, several challenges arise following the providing of PAs remotely and which come in two main issues: the first is to have a scalable and open environment for tutors to design, model practical activities and set it accessible to learners. The second challenge is to have a collaboration and communication environment, with a great degree of realism and to give the students the opportunity to take their practical activities by immersing them as much as possible in a learning situation. These requirements have pushed researchers in education domain to orient themselves towards the 3D environments as multi-user virtual worlds (MUVW) like Second Life, OpenSimulator and others, in order to measure their efficiencies and their capacities to meet the needs of teachers and learners [10][11][12][13][14]. Therefore, a set of 3D simulation solutions in MUVE have emerged like in [15][16]. Even if this approach offers very immersive environments with a high degree of realism, its implementation remains difficult to achieve. As a teacher, this difficulty occurs on four levels, namely to model, animate, interact and share with learners a 3D scene containing objects.

This paper presents a solution for developing practical activities based on OpenSimulator MUVWs, specifically using the open source software OpenSpace 3D [17]. The purpose is to create virtual reality applications by using a simple graphical interface and minimizing the code as much as possible. The proposed solution is a bridge between the 3D scenes created with openSpace 3D and the worlds of OpenSimulator [18] which aims to make the creation of virtual practical activities easier for teachers.

This paper is divided into several sections and is organized as follows: Section II presents the various works that have been done to ensure PAs in e-learning and illustrates a comparative study of the different proposed approaches. In Section III, we present the architecture of the proposed framework as well as it

sub-modules, and particularly the transformation module which is fully described in Section IV. Section V presents a case study to test the proposed framework and thus it shows the results obtained before a discussion and a conclusion section (Section VI).

II. Literature Review and Motivation

Distance education has seen a great evolution as Mari Lahti and her colleagues show in [19]. This development is due to the true potential of the e-learning concept. In fact, distance learning is a very useful training tool for researchers who have problems related to their geographic location or adult learners who have work commitments, family constraints or other obstacles [19]. Contrariwise, we may be confronted with topics that require PAs, especially the courses that refer to the fields of engineering and science [20]. Moreover, experiments can emphasize prominent information or remove confusing details [20].

Several approaches have come to solve the problem of remote practical activities. In this section, we present these solutions and we try to show and discuss their weaknesses.

A. Remote Control Laboratory (RCL)

The principle is to enable different learners to remotely control real equipment using a software interface as in [21][22][23][24]. During the experiment, the results obtained are retrieved and stored in a database, or displayed on the interface that the learner uses to communicate with the remote laboratory.

Fig. 1 illustrates a basic model of RCL based approaches. In remote labs, students manipulate physical apparatus and get real data from physical experiments [20]. This solution is very limited for several reasons:

- The investment to ensure the purchase and the maintenance of the real hardware is always a problem in such cases. On the other hand, if a device fails during testing, the student is obliged to contact the person responsible for maintenance. The presence of a technician is mandatory, and the resumption of the workshop can take hours.
- If this solution offers us the ability to control robots or remote machines, it cannot ensure practical workshops (for example, in chemistry). It can only be used in teaching simple engineering experiments.
- The most prevalent downside is the large number of experiments in relation to practical activities that can run simultaneously. The major problem in education, especially at the university, is the massive number of enrolled students, and therefore the number of required workshops and investment they need.
- This can cause feelings of isolation for the student and hence reduces his/her motivation [21].

B. The use of Videos

To best fit the needs of this problematic and the different limits of the solution for remote control of the real hardware, another proposal has emerged. The idea is to record one version of the practical activity and share it with the learner when it reaches the stage of practice in his learning course [25].

This solution solves, actually, a few of the previous limits, but it puts the learner in a passive learning state, due to the lack of interactivity, and especially for learners who need to be immersed in the learning experience in order to understand the different notions.

C. Virtual Laboratories and Simulators

Currently, either for ‘distance’ or ‘face-to-face’ education, the use of virtual laboratories (VL) experienced a great growth in various disciplines, namely the modeling and simulation of complex systems in physics [26][27], chemistry [11][28], biology [29] [30][31] and even robotics and engineering [32] [33].

This growth is due to several reasons. Indeed, the use of such materials has very significant advantages. In [34], the authors show that VL has several objectives on different levels such as instrumentation, models, equipment, data analysis, design, creativity, psychomotor, security, communication and teamwork.

Nevertheless, these environments are curtailed. A virtual laboratory is a set of virtual objects modeling real objects. Therefore, their extension is very difficult. If the tutor needs to add an object for some reason, he/she must wait for the release of the next version, and there can be no assurance that these processes will fit the expectations of the users.

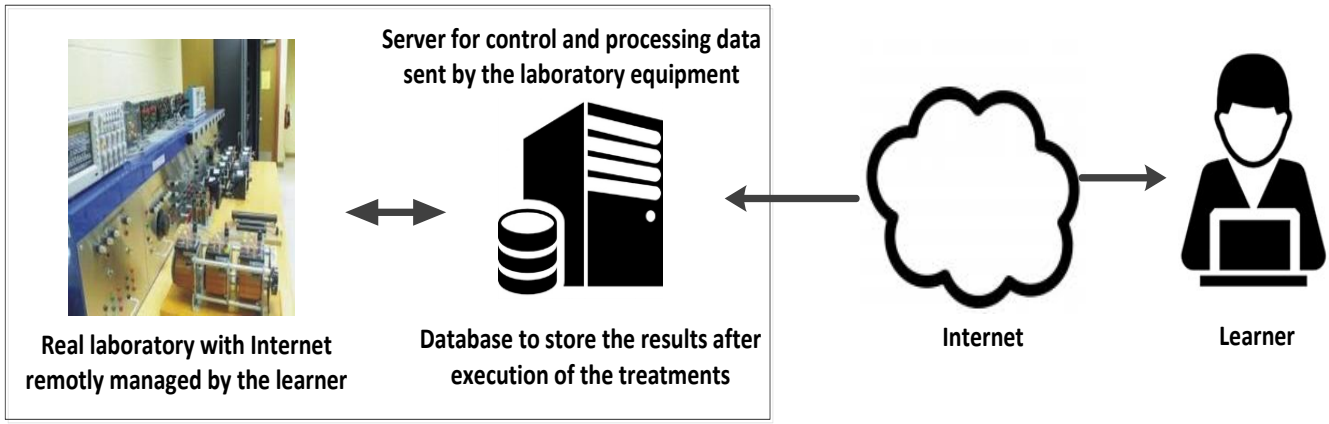

Fig. 1. Remote Control of Real Equipment Model.

D. The Use of Multi-User Virtual Worlds (MUVW)

Another way to ensure simulations, which becomes increasingly adopted [35], is the use of MUVWs. This growth is evidenced by the significant amount of research on this topic and the number of affected areas.

First, a multi-user virtual world, defined in [35], is an online environment, allowing multiple users simultaneous access to virtual resources and contexts where each user is represented by an avatar. Users can interact, communicate and share experiences that may be similar to the real world. Currently there is a large number of MUVW platforms such as Second Life, Open Simulator, Multiverse, and many others for rotating MUVWs.

Today, the features listed in the definition are no longer restricted to MUVWs, but they are redirected to be used in other areas, namely science [36], medicine, engineering [37] and many other disciplines [38].

E. Comparative Study of Existing Approaches

To offer a solution that meets as much as possible the needs of users, a comparison of existing solutions is required. The following table (Table I) illustrates the main differences between them, based on criteria that we considerate pertinent.

It can be seen that VLs share with MUVWs the same strengths in terms of the realization and the learner pedagogical situation. For against, the use of MUVWs leave a great level of immersion as well as the teachers and learners interaction, especially when it comes to 3D virtual worlds.

VLs remain limited to the objectives for which they were developed. Therefore, if the teacher wants to modify the proposed practical activity, s/he is compelled to code (knowing that all the proposed solutions are not open source), which requires the expertise of experts in this kind of programming cases.

TABLE I. COMPARATIVE STUDY OF EXISTING APPROACHES

Solution / ***Criterion***	***RCL***	***Videos***	***VL***	***MUVW***
Realization	Very expensive	Less expensive	Less expensive	Less expensive
Maintenance	Very expensive	No cost	--	--
Level of immersion	Unavailable	Unavailable	Limited	Available
Evolution	--	--	Not always possible	Possible
Cost	Very expensive	--	Expensive	Less expensive
Learner pedagogical situation	Liability	Liability	Asset	Asset
Interaction	Unavailable	Unavailable	Limited	Excellent

III. THE PROPOSED APPROACH

A. Presentation

The primary focus in this work is to design and implement a computer system addressing the issue of simulation by providing an approach for the preparation of virtual PAs and meeting the limits of existing approaches. At this level, several questions arise:

- How can the teacher create practical activities while having an extensible environment?
- How to create interactive objects, upload them and send them to students, knowing that not all teachers are computer scientists or either have programing skills?
- Is there a way to realize solutions and contextual factors in the different workshops between students?

In this section, we present our project: a simulation system composed of several modules that we consider important to model, animate and realize a practical activity. This section is divided into several parts, the first describes the overall architecture of the proposed system, and the second presents the approach used to create and realize a practical activity, while the third part explains the architecture's modules and the technologies used. Finally, the fourth part details an important module of the solution, which is the transformation module and represents the proposed system base.

B. Global Architecture

As we have mentioned, the framework we propose is divided into several modules, as shown in Fig. 2. The first module is a 3D object design tool. The second is a scripting tool for animating and making the interaction with the objects provided by the graphic designerThe transformation module manages the conversion of the language and formats generated after the teacher animates the objects of PAs to the language and format supported by the MUVW, this module is practically the cornerstone of this project.

The Fig. 2 shows also the actors involved in the system in its entirety. On one hand, there is the designer who, with his/her expertise, helps to design the PA. In a second hand, there is the teacher, as an essential actor in a teaching situation. The system must provide him/her with a set of features allowing him/her to prepare, monitor and control PAs. Eventually, the student is the targeted actor and the most important one. The system must offer him/her a sophisticated environment facilitating the communication with the other students, in addition to his/her interaction with the workshop and the objects. Thus, it will help assimilate easily the different concepts applied in a workshop.

C. The System Modules and Technology

1) 3D modeling tools: The main function of this module is to model 3D objects to animate them later in OpenSpace3D. In fact, a specific tool does not limit us, any 3D modelling software can do the trick, we quote Blender, Maya, 3D Max, etc.

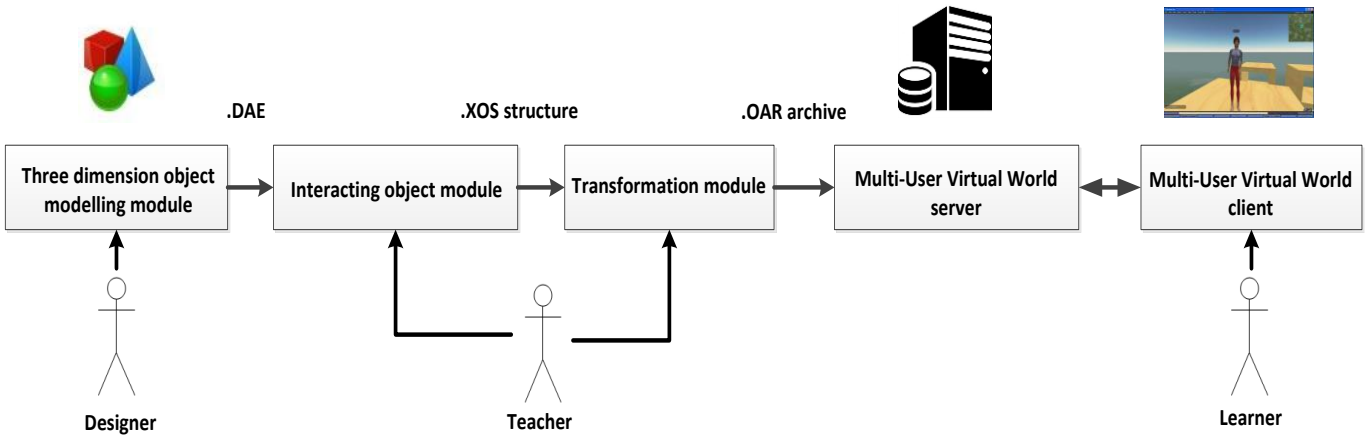

Fig. 2. Global Architecture.

2) Scripting environment of practical activities: It is the main tool with which the teacher, with the help of a graphic designer, can prepare practical workshops, and animate events (e.g., click, and double click). This module is the OpenSpace 3D software, which is an open source platform for developing Virtual and Augmented Reality projects. It allows full build of interactive 3D scenes graphically rich without entering any line of code. On the other hand, it also has many functions ready for use. Achieving an OpenSpace3D implementation therefore is integrating different functions and defining their reciprocal interactions.

OpenSpace 3D is based on plugIt principle to animate 3D objects. PlugIt is a function package already developed with Scol language [39] and a very simple way to animate 3D objects. The tool offers a sophisticated interface, divided into 3 parts (Fig. 3): the first one lists the imported 3D objects in the project, the second part shows the object's scene in 3D and the last part is where the user animates the objects with PlugIts just by using links between the event and the action. A link is a connection between an event function and an action of another or the same function. After the animation of the scene, the user saves his/her project under XOS format or s/he can import it towards a Scol project or a standalone application.

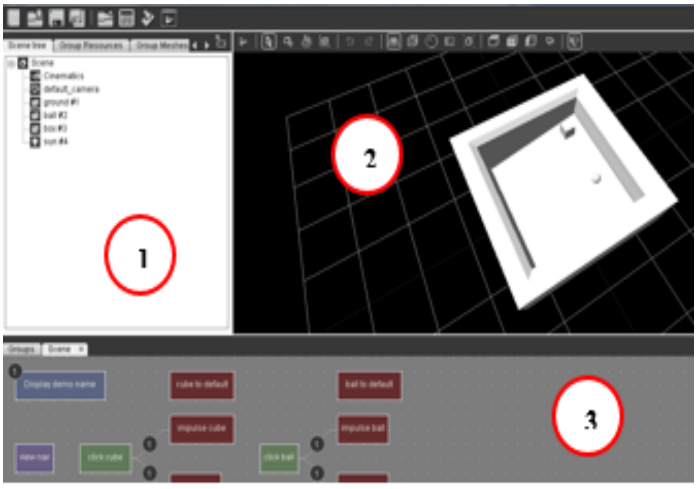

Fig. 3. OpenSpace 3D Environment.

OpenSapce 3D is not only limited to the creation of virtual reality applications, but also offers a set of Plugit dedicated to the management of augmented and mixed reality. It also offers the possibility of use and integration of several sensors, for example Leap Motion and Kinect Xbox. It can facilitate the development of applications and games which can be oriented to the educational area.

3) MUVW Module: This module will, to some extent, play the role of virtual laboratory. It represents the environment where the learner will make his/her PAs and communicate with other users whether they are students or teachers. The choice was made on the OpenSimulator platform[40], an open source project under the BSD license. It aims to develop a functional platform for virtual worlds capable of supporting multiple clients and servers, all in a heterogeneous grid structure. It allows great freedom concerning the development of specific applications, hosting and controlling overall costs for economic and efficient projects. This is partly justified, by the free license contrary to proprietary solutions.

Furthermore the interface (3D client browsers) already incorporates comprehensive tools for creating content accessible to neophytes (3D objects from simple primitive forms, scripts for programming, management of the 3D environment, etc.). Furthermore, design professionals who are familiar with softwares such as Blender, Maya, and 3DS Max, etc., can also incorporate 3D content created from these professional softwares. Finally, some collaboration tools are available in Open Simulator like Vivox, the leader in voice conference, Shared Media that allows a simple integration of multiple, rich and diverse media (web pages, video, images, etc.). These tools bring Open Simulator to an unparalleled level of functionality compared to other solutions.

4) Transformation module: This module is the processing interface that adapts 3D objects animated by the practical workshops' preparation module (OpenSpace 3D) to the OAR format used by Open Simulator. This transformation module, fully explained in the next section, exploits the transformation rules and the code generation to LSL (Linden Scripting language), the animated language adopted by Open Simulator to animate 3D objects.

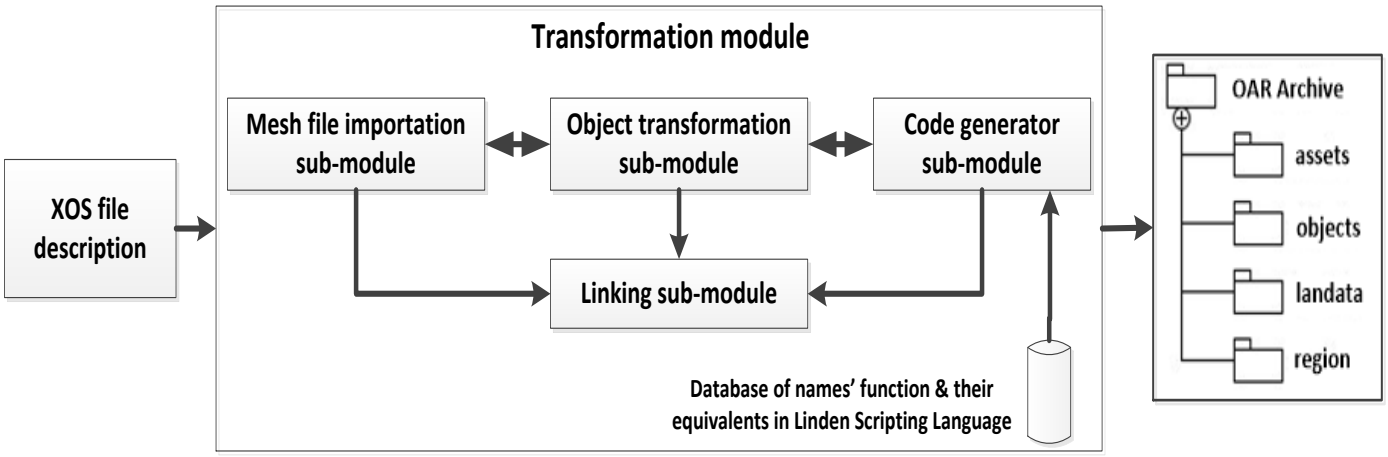

Fig. 4. Transformation Module Architecture.

D. Transformation Module Details

1) Architecture: For each scene of objects, OpenSpace3D generates an XML file with XOS extension that contains a description of the entire scene, events, and the location of 3D objects on the hard disk. The import unit searches the XML list files of 3D objects used in the practical workshop, and then it loads the objects in the OpenSimulator objects database. The transformation engine provides the correspondence between the activities defined in the XML file and generates the relevant objects and LSL Script based on defined transformation rules, and at the end, it loads the scripts on the server. The Fig. 4 illustrates the transformation mechanism from the XOS format into the OAR archive format.

We have segmented the transformation of the XOS files into the OAR format in several parts. The first step is to import the necessary resources for the construction of 3D objects and thus of the scene. The second step is searching for the positioning information of each object on the stage and its characteristics. The third step is to transform the animations by searching in the database for the functions in LSL equivalent to those mentioned in the XOS file. The last step is to ensure the link between the objects, their characteristics and the animations (interactions).

2) Meta-model of XOS: After animating 3D objects of practical activity in OpenSpace 3D, the teacher saves the entire scene as a single compressed XML file with the XOS extension. The main role is to describe all objects in the scene, cameras, reference objects and their positions in the scene on one side, and the different events and own animation of each object in the other side. The Fig. 5 presents a part from an example of the XOS file.

We are based on several examples developed under Open Space 3D to propose a general meta-model of the XOS description. The Fig. 6 shows a simplified Meta model of an XOS file.

```
<project author="OpenSpace+3D+Objects+group+based+on+OgreMax+Scene+Exporter+%28www%2eogremax%2ecom%29" formatVersion="1%2e22%2e0">
    <scene py="1" px="3">
        <resources>
            <resource path="assets%2fmodels%2fconverted%2fchute%5flibre%5fcollada%2fmaterials%2fchute%5flibre%5fcollada%2ematerial" type="material"/>
            <resource path="assets%2fmodels%2fconverted%2fchute%5flibre%5fcollada%2fmeshes%2fchute%5flibre%5fcollada%5ftext%5f0042%2emesh" type="mesh"/>
                                                    ⋮
    <plugins>
        <plugin name="object+click" source="objects%2fobjectclick%2fobjectclick%2exml">
            <instance py="1" px="9" name="click+on+the+0%2e2+value" comment="">
                <param name="object"><![CDATA[14.chute_libre_collada_Text19]]></param>
                <param name="material"><![CDATA[chute_libre_collada_Material_0010]]></param>
                <param name="enablemat"><![CDATA[1]]></param>
                <param name="cursor"><![CDATA[1]]></param>
                                                    ⋮
    <mesh visibilityFlags="0xFFFFFFFF" nbLODlevel="0" renderQueue="50" renderingDistance="" indexMaterials="false" receiveShadows="false" castShadows=
        <subentities>
            <subentity defaultMaterial="chute%5flibre%5fcollada%5fMaterial%5f0010" materialName="chute%5flibre%5fcollada%5fMaterial%5f0010" index="0"/
        </subentities>
        <scale z="3%2e089895" y="0%2e484955" x="0%2e484955"/>
        <position z="3%2e104373" y="0%2e282935" x="%2d2%2e195202"/>
        <rotation qw="1%2e0" qz="0%2e0" qy="0%2e0" qx="0%2e0"/>
```

Fig. 5. Example of an XOS File.

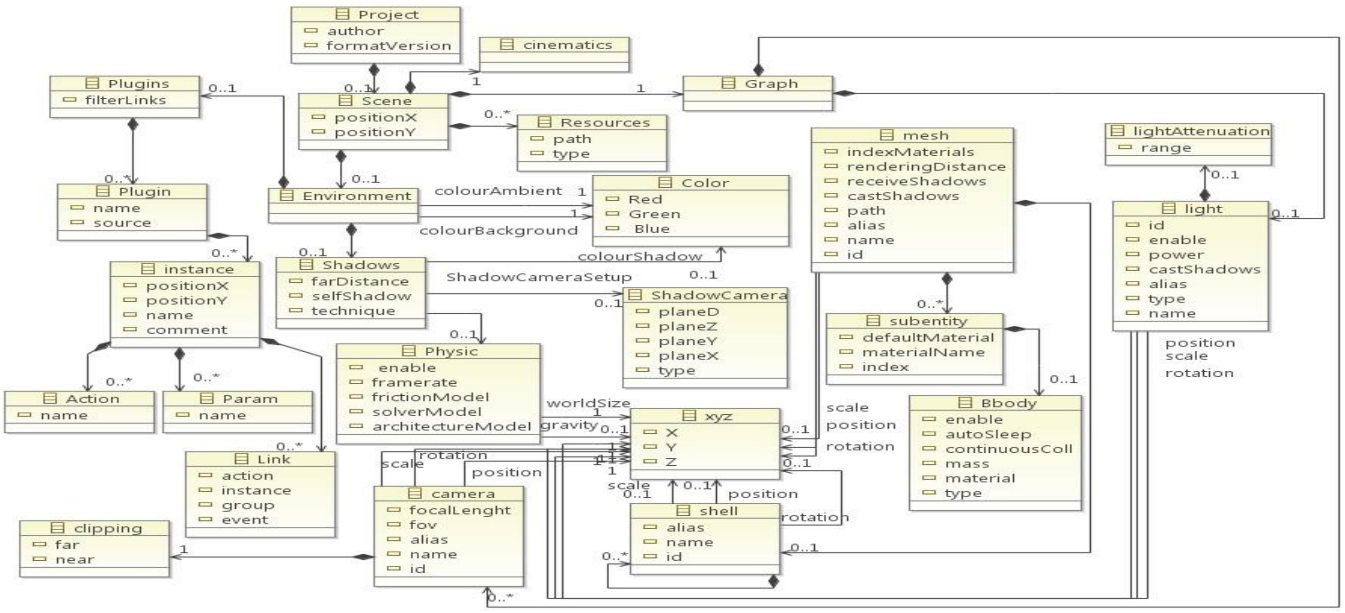

Fig. 6. Simplified Meta-Model of XOS.

3) Meta-model of OAR format

a) The OAR model: The OpenSimulator Archive (OAR) function as it is introduced in the official web site of OpenSimulator has existed since OpenSimulator 0.5.9. The facility does a similar job to load-xml2/save-xml2 and it saves prims so that they can be reloaded later.

However, OpenSimulator (OAR) archives go further in this direction, since they can back up all the inventory data needed to fully restore terrain, region parcel data, object textures and their inventories when they are loaded on a completely different system using a different asset database.

An OAR file is a "gzipped" tar file in the original Unix tar format. This can be extracted and created with standard tools. The Fig. 7 shows the OAR structure and components.

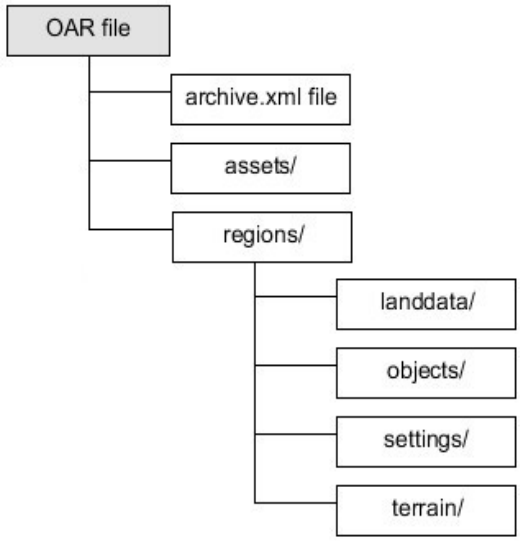

Fig. 7. OAR Format.

As we can see in Fig. 7, the OAR format is a set of directories; each one has a specific role. In our approach, OAR is the exchange format between the teacher and different learners. As following, a global description of the OAR format:

- Archive.xml: This is the archive control file. It contains a major and minor version number, to allow compatibility with future format changes.
- Assets/: This directory contains all the assets in the archive. The assets for all the regions are stored in the same directory because assets are often shared. Each filename has the following format <uuid>_<asset type>.<asset extension>. The uuid section must always be present and form a valid uuid - it is used directly for that asset. The asset type and asset extension are used to identify the type of asset and the asset extension allows the asset to be associated with different editors on platforms such as Windows. For instance, a script will always have the asset type and extension script.lsl. A full list of asset types and extensions can be found in the file.
- Landdata/ : This directory contains all the parcels in the region. Information for each parcel is stored in a separate file in XML format.
- Objects/ : Each individual file is an object in the region (where an object [linkset] can be composed of many prims). The file format used is OpenSim's XML2 format. Each filename has the following structure by default.
- Settings/ : This contains the region settings information for the region in XML format. The filename will be the same as the region name.
- Terrains/ : This contains the terrain file for the region, stored in RAW format. The filename must end with .r32.

```
<SceneObjectGroup>
    <SceneObjectPart xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
        <AllowedDrop>false</AllowedDrop>
                    ⋮
        <GroupPosition>
            <X>126.343</X>
            <Y>125</Y>
            <Z>29.08239</Z>
        </GroupPosition>
        <OffsetPosition>
            <X>0</X>
            <Y>0</Y>
            <Z>0</Z>
        </OffsetPosition>
                    ⋮
        <Color>
            <R>0</R>
            <G>0</G>
            <B>0</B>
            <A>255</A>
        </Color>
                    ⋮
        <Shape>
            <ProfileCurve>5</ProfileCurve>
            <TextureEntry>iVVnRyTLQ+2SC0fK7RVGXwAAAAAAAAAAIEEAAAAgQQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA</TextureEntry>
            <ExtraParams>AA==</ExtraParams>
            <PathBegin>0</PathBegin>
            <PathCurve>32</PathCurve>
            <PathEnd>0</PathEnd>
```

Fig. 8. Example of Object Presentation in OpenSimulator using XML Model.

To identify equivalences between XOS model and the 3D objects OpenSimulator, we have also created a meta-model.

b) Object's model on OpenSimulator: The most important in this first stage is to export objects with animation script from XOS format to OAR archive. Therefore, we focused on the objects directory to model 3D object and on the assets directory to create script animation and affect each one to its 3D object. Firstly, we developed a meta-model of how 3D objects are modelled on OAR format. Each 3D object is defined with a file in the objects directory.

The Fig. 8 shows a part from an example of object presentation in OpenSimulator using XML model.

We are based on several examples developed under Open Space 3D to propose a general meta-model of the XML 3D object description in OpenSimulator. The Fig. 9 shows the schematic simplified representation of a 3D object in an OpenSimulator's scene.

Each object is a "SceneObjectGroup" in the representation's file, this SceneObjectGroup is a set of SceneObjectPart, each SceneObjectPart has <x,y,z> coordinates. If SceneObjectGroup is a set of many mesh objects, a SceneObjectPart is defined in this case as a set of SceneObjectPart which is, in turn, defined in an OtherPart tag.

c) Linden Scripting Language (LSL): Is the programming language used by residents of OpenSim. LSL has a syntax similar to C language and allows objects to control the behavior of in-world objects of Second Life from the Internet via email, XML-RPC, and most recently, HTTP requests.

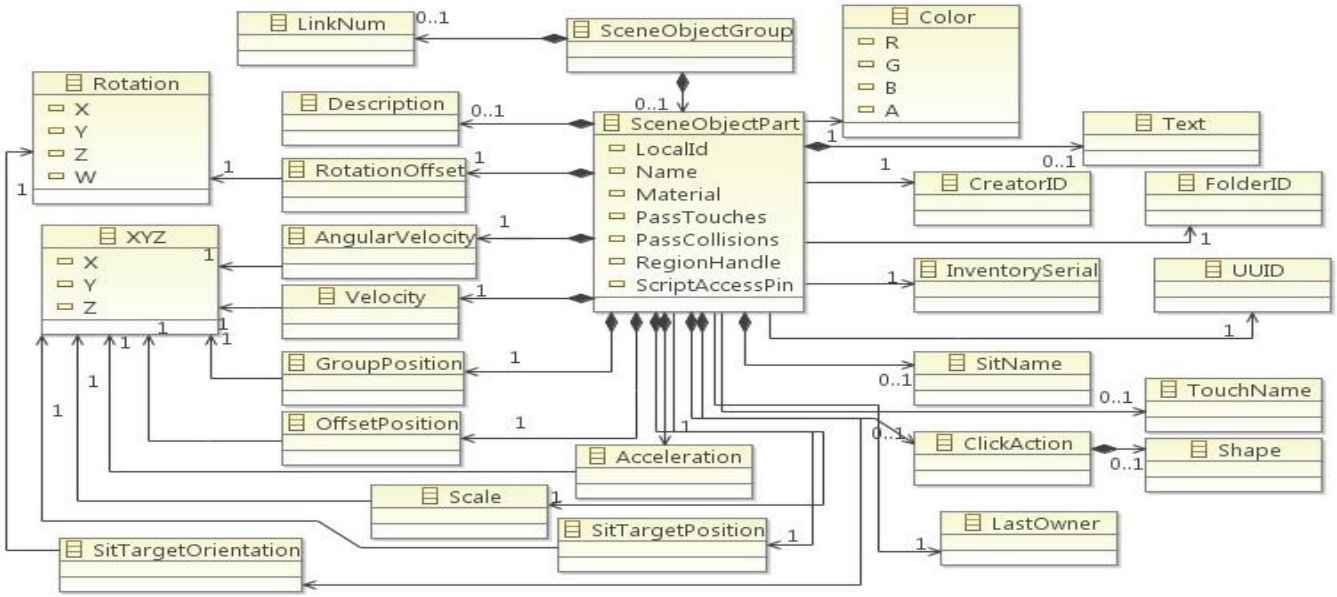

Fig. 9. Simplified Meta-Model of Object Presentaion in OpenSimulator Environment.

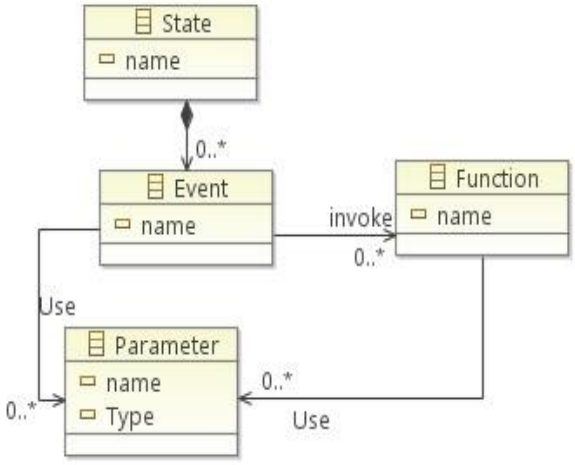

Fig. 10. Simplified Meta-Model of LSL Language Structure.

Linden Scripting Language is a state-event driven scripting language, a type of finite state machine. A script consists of variables, function definitions, and one or more named states. Each state contains a description of how to react to events that occur while the program is within that state. The system sends to the script, such as timers, movement, chat (from other agents), email, and collisions (with objects in the virtual world). Scripts can change most aspects of the state of the object and communicate with other objects and agents. As soon as a script is added to an object, and turned on, it begins to execute. The Fig. 10 shows a simplified LSL's meta-model.

4) Transformation rules: The transition from XOS to OAR model requires firstly a correspondence between the XOS and OAR elements. This correspondence is divided into two parts: the first one is for the transformation of 3D objects and their positions on the scene (Table II); the second part is the transformation of different animation's scripts (Table III).

TABLE II. XOS TO OAR OBJECT FORMAT

XOS	**OAR objects format**
Shell	SceneObjectGroup
Mesh	A file with .llmesh extension in the asset directory
Xyz	Xyz
Color	Color
Scale	Scale
Position	GroupPosition
Rotation	RotationOffset

TABLE III. PLUGIT TO LSL FUNCTION

Plugit	**LSL**
Action	Function
Param	Parameter
Event	Event

```
default
{
    touch(integer num_detected)
    {
        llSetPos(llGetPos() + <0,0,2>);
    }
}
```

Fig. 11. Example of Generated LSL Code.

Animations in XOS are described in XML format, and grouped into several plugins. Although the components of each plugit differ from one plugit to another (especially at the number of parameters) and depending on the desired animation, there is a standard part that describes the function of the animation and the event that triggers it. For example, the description of the plugin of moving an object by clicking on the left mouse button will be as follows:

```
<link action="Goto" instance="ball+goto+0%2e2" group="Scene" event="LeftClick"/>
```

Each attribute has a meaning:

- Action: the name of the function to be called.
- Event: the name of the event that triggers the action.
- Instance: The name of the instance that contains the parameters needed to call the "GoTo" function, and which in turn, has a particular description containing the new position (x, y, z) of the object after executing the "Goto ".

By respecting the transformation rules proposed in Table III, we obtain an LSL file containing the code of the LSL animation (Fig. 11):

The generated LSL code consists of several blocks. The first block is "default", which represents the default state of the object. An object can have several states depending on its nature, the person object for example can be on or sitting, the light object can be on or off. The second block "touch" represents the event that will change the position of the object (in our case it is the event click on the right button of the mouse). The "LLsetPosition" function is the LSL function that changes the position of the object according to the cardinalities (x, y, z).

The generated script is placed in the "assets" folder in the OAR archive, and it is linked to the "ball" object via its unique name. Each object in "OAR" format has a unique identifier, and each script is attached to its object with the same name as the object in question.

IV. CASE STUDY

A. Presentation, Protocol, and Objective of the Workshop

Mechanics remain a special educational area, this comes from the fact that the majority of the notions, introduced in its courses, need visual experiments and practical activities so that a learner can assimilate these notions and understand their usefulness.

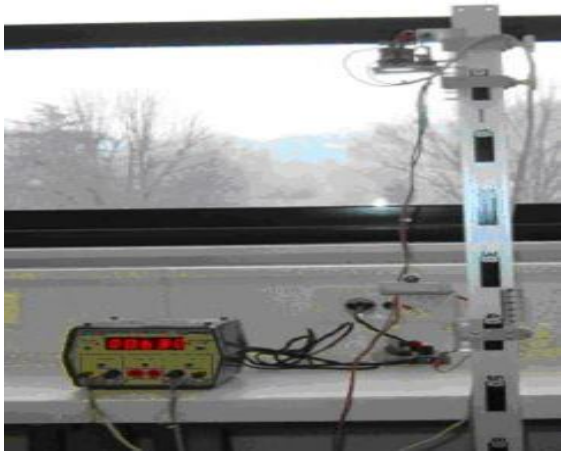

Fig. 12. Practical Activity with Real Equipment.

Mechanics remain a special educational area, this comes from the fact that the majority of the notions, introduced in its courses, need visual experiments and PAs so that a learner can assimilate these notions and understand their usefulness. The case presented in this section of the paper illustrates the study of a ball free fall where the ball is subject to a single force, its weight, and the other forces are neglected. The main objective is to determine the relationship between the ball's height and the free fall time. The learner takes a ball with fixed mass and tries to change the height in order to launch the ball. An electronic clock measures the travel time (t) of the ball from its origin. The learner repeats the experimentation many times by changing the height of the ball (h), notes the fall time on the clock, calculates and notes the quotient h/t^2.

This experiment generally requires a hardware device called a free fall device (Fig. 12). The latter includes several parts, namely a digital counter for calculating the falling time, a micro-magnet which retains the ball in the starting position, a switch for starting the time measurement, and finally a plate on which the ball strikes in order to stop the time measurement.

B. Virtualization Steps

The virtualization of the workshop described above has undergone several steps. First, we have developed a list of minimal requirements from hardware devices. There are the ruler with which the learner can choose the height of the ball, a ball and a display showing the time of the free fall.

The second step consists in modeling the various devices in three dimensions. To do so, we used the "blender" software as shown in Fig. 13.

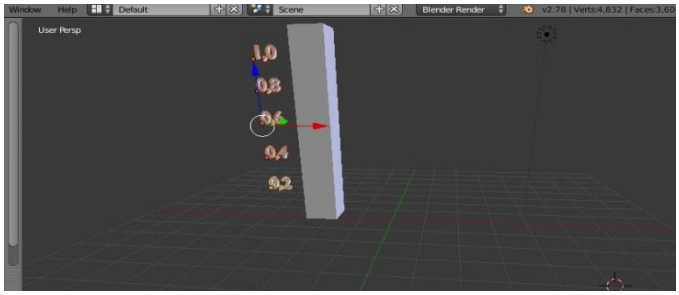

Fig. 13. Creation of Needed Objects in Blender Software.

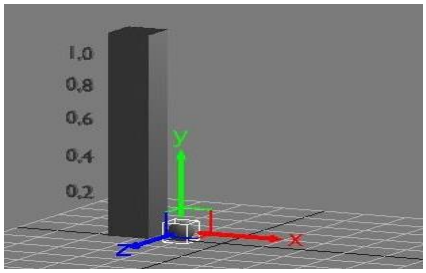

Fig. 14. Importation of All Objects in OpenSpace 3D Software.

Once the objects are created, Blender offers the possibility of exporting the scene to several formats. We chose "collada" format which is the standard format for exchanging 3D objects.

Step three consisted of programming the interaction of the virtual devices with the users in order to give them the opportunity to simulate the practical workshop and to collect information (the displayed time of the free fall). For this, we used the OpenSpace3D software. We have imported 3D objects and then created a scene. We placed the different objects in their initial positions and animate the objects so that if the learner clicks on a number on the ruler (a chosen height), the ball changes its position and moves to the opposite of the mentioned height, see (Fig. 14).

Once the user clicks on the ball, the ball falls freely to the ground and the time counter gives the duration of the fall. To ensure this interaction, we have called the various plugIts of the OpenSpace3D software (Fig. 15). The latter offers a variety of animation plugIts allowing the change of the position of an object, rotation, or even treating object collision without coding.

It remains to point out that the 3D engine used in OpenSpace3D offers the exploitation of physics laws by giving the possibility to parameterize gravity, mass and many other options by proposing a plugit for that.

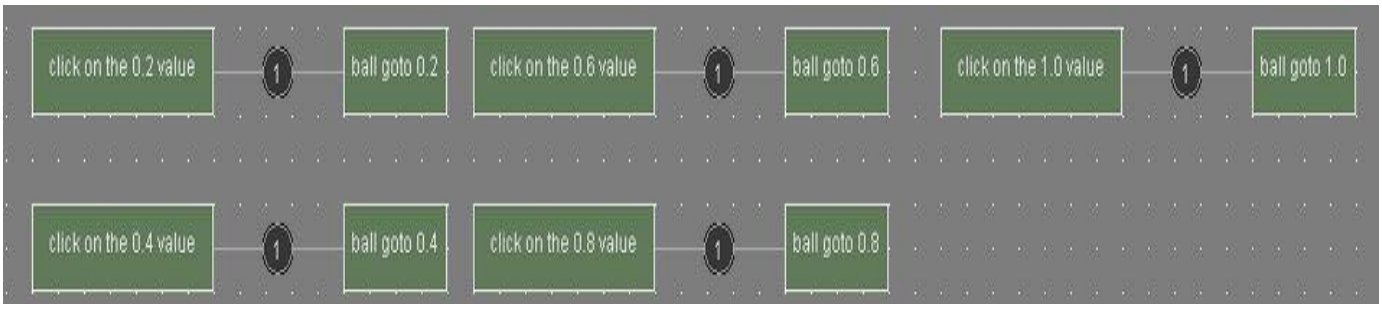

Fig. 15. Animating Objects in OpenSpce 3D Software.

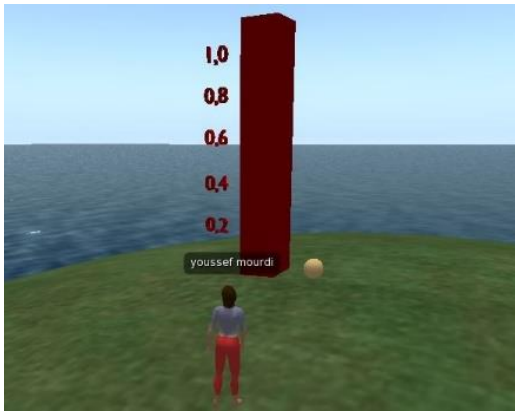

Fig. 16. Obtained View after OAR Importation.

The last step was to take the generated OpenSapce3D file with the XOS extension after setting up the interactions and to import it into our transformation module to get the final result, which is an OAR package containing the files, mesh, materials, and all the components of our PA in order to import it on the OpenSimulator platform. We notice that the objects are imported and placed correctly on the OpenSimulator scene.

C. Results

We have imported the resulting archive of the transformation module to OpenSimulator, using the commands of OpenSimulator console to load the OAR archive. The result is very satisfying in terms of the devices presentation (Fig. 16).

The first challenge was to import a scene of objects from an XOS description to an OAR structure. In accordance with the transformation rules described in Section 4, we obtained the following OAR structure (Fig. 17).

As regards to the layout of the objects on the scene, the transformation module was based on the coordinates (x, y, z) described in the XOS files. The transformation module imported the necessary meshes and generated an XML file for each object that described it in the "assets" folder of the OAR archive (Fig. 18).

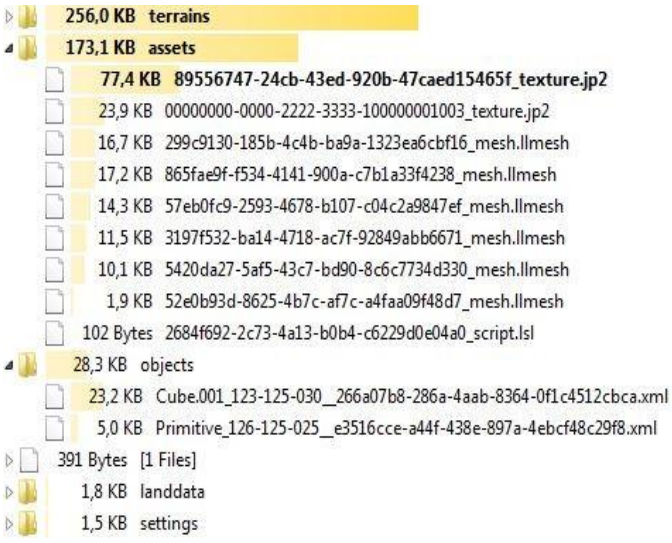

Fig. 17. The Generated OAR File.

```
<GroupPosition>
    <X>126.343</X>
    <Y>125</Y>
    <Z>29.08239</Z>
</GroupPosition>
<OffsetPosition>
    <X>0</X>
    <Y>0</Y>
    <Z>0</Z>
</OffsetPosition>
<RotationOffset>
    <X>0.2008292</X>
    <Y>0.677988</Y>
    <Z>-0.2008292</Z>
    <W>0.6779879</W>
</RotationOffset>
```

Fig. 18. Position of the Group Objects.

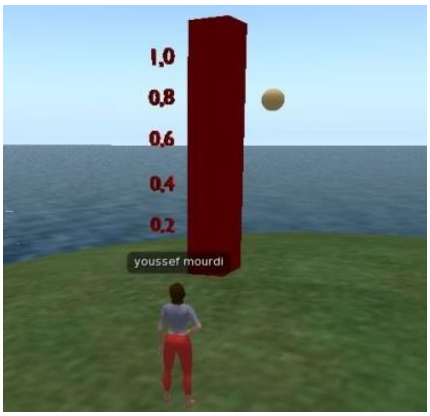

Fig. 19. Interaction between the Learner and the Object.

The second challenge we had was to keep the objects interacting with the users. To deal with this, the transformation module generated the necessary LSL files containing the calls to functions already programmed in LSL (Fig. 11).

When a user (learner) touches a number (selected height), the ball moves to the chosen height as seen in Fig. 19.

V. Conclusion

E-learning is a new field that has its roots in several sectors such as education and training, open distance learning, knowledge management, quality etc. National and international organizations, academic institutes and experts focus on analysing the success of e-learning and on how best to develop new approaches and structures to ensure ever better quality. The integration of practical activities is a very important task and is the cornerstone of a high quality education.

The practical activities of e-learning are progressing rapidly, taking advantage of the latest technologies and using various principles of pedagogy and organization. Today, their integration is characterized by a high level of innovation, experimentation and research of the most adapted solutions, such as remote control of real equipment, videos or virtual laboratories, that require more improvements because they do not really adapt to the expectations and needs of the learners or to the habits of teachers.

A common framework therefore needs to be put in place to both ensure a realistic open environment accessible to all and to provide learners with a high level of immersion while

performing their practical activities. This was the purpose of this paper where we give teachers, the opportunity to virtualize the practical activities, and to use the forces of virtual worlds in terms of communication, immersion, and the degree of realism for pedagogical purposes especially in the fields of distance learning and training.

The main objective of our research was to develop a transformation module allowing a user to import scenes of interactive objects from the OpenSpace3D software to a format recognized and importable in the virtual worlds of OpenSimulator. This transformation module helped in transforming a scene of animated objects using OpenSpace3D to an OAR archive importable on the virtual worlds of OpenSimulator. This proposal will allow teachers and especially non-computer scientists to create virtual workshops and make the various devices interactive without coding anything; this is thanks to OpenSpace3D software.

The results obtained in this paper are very satisfying. The limitations, which we must remedy later, are generally at the level of the transformation module development. The functions of OpenSimulator are not always the same adopted by OpenSpace3D in terms of signature and parameters. Consequently, future works will focus on improving the architecture proposed in this paper, as well as the search for integration of an intelligent interface to ensure the best transformations.

REFERENCES

[1] K. Robins and F. Webster, “The Virtual University?,” Virtual Univ. Knowledge, Mark. Manag., pp. 3–19, 2002.

[2] W. R. Watson and S. L. Watson, “What are learning management systems, what are they not, and what should they become?,” TechTrends, vol. 51, no. 2, pp. 28–34, 2007.

[3] L. D. Feisel and A. J. Rosa, “The role of the laboratory in staff development.,” J. Eng. Educ., vol. 94, no. 1, pp. 121–130, 2005.

[4] C. Coti, J. V. Loddo, and E. Viennet, “Practical activities in network courses for MOOCs, SPOCs and eLearning with Marionnet,” 2015 Int. Conf. Inf. Technol. Based High. Educ. Training, ITHET 2015, 2015.

[5] M. Casini, D. Prattichizzo, and A. Vicino, “The automatic control telelab: A remote control engineering laboratory,” Proc. 40th Ieee Conf. Decis. Control. Vols 1-5, no. February, pp. 3242–3247, 2001.

[6] C. Lazar and S. Carari, “A remote-control engineering laboratory,” IEEE Trans. Ind. Electron., vol. 55, no. 6, pp. 2368–2375, 2008.

[7] D. Hercog, B. Gergic, S. Uran, and K. Jezernik, “A DSP-Based Remote Control Laboratory,” IEEE Trans. Ind. Electron., vol. 54, no. 6, pp. 3057–3068, 2007.

[8] E. Guimarães et al., “REAL: A Virtual Laboratory for Mobile Robot Experiments.,” IEEE Trans. Educ., vol. 46, no. 1, p. 37, 2003.

[9] L. Xu, D. Huang, and W. Tsai, “V-lab: a cloud-based virtual laboratory platform for hands-on networking courses,” Proc. 2012 ACM Conf. Innov. Technol. Comput. Sci. Educ., pp. 256–261, 2012.

[10] J. Lima, L. Morgado, B. Fonseca, P. Martins, and H. Paredes, “Effectiveness of virtual world timers in educational physics simulations,” SLACTIONS 2011 - Res. Conf. Second Life world Life, imagination, Work using metaverse platforms, November, 2011, 2011.

[11] B. Dalgarno, “The potential of virtual laboratories for distance education science teaching: Reflections from the development and evaluation of a virtual chemistry laboratory,” Proc. Aust. Conf. Sci. Math. Educ. (formerly UniServe Sci. Conf., vol. 9, pp. 90–115, 2012.

[12] K. Andreas, T. Thrasyvoulos, D. Stavros, and P. Andreas, “Collaborative learning in OpenSim by utilizing Sloodle,” in 6th Advanced International Conference on Telecommunications, AICT 2010, 2010, pp. 90–95.

[13] T. Ritzema and B. Harris, “The Use of Second Life for Distance Education,” J. Comput. Sci. Coll., vol. 23, no. 6, pp. 110–116, 2008.

[14] C. Allison et al., “Growing the Use of Virtual Worlds in Education: an OpenSim Perspective,” EiED 2012 Proc. 2nd Eur. Immersive Educ. Summit, pp. 1–13, 2012.

[15] M. Rico, J. Ramirez, D. Riofrio, M. Berrocal-Lobo, and A. De Antonio, “An architecture for virtual labs in engineering education,” IEEE Glob. Eng. Educ. Conf. EDUCON, pp. 210–214, 2012.

[16] M. J. Callaghan, K. McCusker, J. L. Losada, J. Harkin, and S. Wilson, “Using game-based learning in virtual worlds to teach electronic and electrical engineering,” IEEE Trans. Ind. Informatics, vol. 9, no. 1, pp. 575–584, 2013.

[17] “OpenSpace 3D.” [Online]. Available: http://www.openspace3d.com/. [Accessed: 30-Jan-2017].

[18] R. González Crespo, R. F. Escobar, L. Joyanes Aguilar, S. Velazco, and A. G. Castillo Sanz, “Use of ARIMA mathematical analysis to model the implementation of expert system courses by means of free software OpenSim and Sloodle platforms in virtual university campuses,” Expert Syst. Appl., vol. 40, no. 18, pp. 7381–7390, 2013.

[19] M. Lahti, H. Hätönen, and M. Välimäki, “Impact of e-learning on nurses’ and student nurses knowledge, skills, and satisfaction: A systematic review and meta-analysis,” Int. J. Nurs. Stud., vol. 51, no. 1, pp. 136–149, 2014.

[20] R. Heradio, L. De La Torre, D. Galan, F. J. Cabrerizo, E. Herrera-Viedma, and S. Dormido, “Virtual and remote labs in education: A bibliometric analysis,” Comput. Educ., vol. 98, pp. 14–38, 2016.

[21] C. A. Jara, F. A. Candelas, S. T. Puente, and F. Torres, “Hands-on experiences of undergraduate students in Automatics and Robotics using a virtual and remote laboratory,” Comput. Educ., vol. 57, no. 4, pp. 2451–2461, 2011.

[22] E. Fabregas, G. Farias, S. Dormido-Canto, S. Dormido, and F. Esquembre, “Developing a remote laboratory for engineering education,” Comput. Educ., vol. 57, no. 2, pp. 1686–1697, 2011.

[23] M. Stefanovic, V. Cvijetkovic, M. Matijevic, and V. Simic, “A LabVIEW-based remote laboratory experiments for control engineering education,” Comput. Appl. Eng. Educ., vol. 19, no. 3, pp. 539–549, 2011.

[24] C. Terkowsky, C. Pleul, I. Jahnke, and A. E. Tekkaya, “Tele-operated laboratories for online production engineering education platform for e-learning and telemetric experimentation (PeTEX),” Int. J. Online Eng., vol. 7, no. SUPPL., pp. 37–43, 2011.

[25] V. Fernandez et al., “‘Low-Cost Educational Videos’ for Engineering Students: a new Concept based on Video Streaming and YouTube Channels,” Int. J. Eng. Educ., vol. 27, no. 3, pp. 518–527, 2011.

[26] G. Quesnel, R. Duboz, and É. Ramat, “The Virtual Laboratory Environment - An operational framework for multi-modelling, simulation and analysis of complex dynamical systems,” Simul. Model. Pract. Theory, vol. 17, no. 4, pp. 641–653, 2009.

[27] Y. Daineko and V. Dmitriyev, “Software module ‘Virtual Physics Laboratory’ in Higher Education,” in 8th IEEE International Conference on Application of Information and Communication Technologies, AICT 2014 - Conference Proceedings, 2014.

[28] C. Tüysüz, “The effect of the virtual laboratory on students’ achievement and attitude in chemistry,” Int. Online J. Educ. Sci., vol. 2, no. 1, pp. 37–53, 2010.

[29] D. Raineri, “Virtual laboratories enhance traditional undergraduate biology laboratories,” Biochem. Mol. Biol. Educ., vol. 29, no. 4, pp. 160–162, 2001.

[30] S. Diwakar, K. Achuthan, and P. Nedungadi, “Biotechnology virtual labs- integrating wet-lab techniques and theoretical learning for enhanced learning at universities,” in DSDE 2010 - International Conference on Data Storage and Data Engineering, 2010, no. February, pp. 10–14.

[31] S. Diwakar, K. Achuthan, P. Nedungadi, and B. Nair, “Biotechnology Virtual Labs : Facilitating.” 2012.

[32] T. Mohsen-Torabzadeh, Z. Muhammed Hossain, P. Fritzson, and T. Richter, “OMWeb – Virtual Web-based Remote Laboratory for

Modelica in Engineering Courses," in Proceedings of the 8th International Modelica Conference, 2011, pp. 153–159.

[33] V. Potkonjak, M. Vukobratovi??, K. Jovanovi??, and M. Medenica, "Virtual Mechatronic/Robotic laboratory - A step further in distance learning," Comput. Educ., vol. 55, no. 2, pp. 465–475, 2010.

[34] M. Stefanovic, "The objectives, architectures and effects of distance learning laboratories for industrial engineering education," Comput. Educ., vol. 69, no. September 2014, pp. 250–262, 2013.

[35] J. Defazio, T. Faas, and R. Finch, "Building multi-user virtual worlds," Proc. CGAMES'2013 USA, pp. 132–137, 2013.

[36] D. J. Ketelhut, B. C. Nelson, J. Clarke, and C. Dede, "A multi-user virtual environment for building and assessing higher order inquiry skills in science," Br. J. Educ. Technol., vol. 41, no. 1, pp. 56–68, 2010.

[37] A. Pinheiro et al., "Development of a mechanical maintenance training simulator in OpenSimulator for F-16 aircraft engines," Entertain. Comput., vol. 5, no. 4, pp. 347–355, 2014.

[38] M. B. Ibáñez, J. J. García, S. Galán, D. Maroto, D. Morillo, and C. D. Kloos, "Multi-user 3D virtual environment for Spanish learning: A wonderland experience," in Proceedings - 10th IEEE International Conference on Advanced Learning Technologies, ICALT 2010, 2010, pp. 455–457.

[39] "Scol programming langage." [Online]. Available: http://redmine.scolring.org/projects/scol/wiki/Scol_Language_particular_syntax. [Accessed: 30-Oct-2017].

[40] "OpenSimulator." [Online]. Available: http://opensimulator.org/wiki/Main_Page. [Accessed: 30-Oct-2017].