An Architecture for Virtual Labs in Engineering Education

Educational Innovation Project at UPM

Mariano Rico

Computer Science Department
Universidad Autónoma de Madrid
Spain
mariano.rico@uam.es

Jaime Ramírez[1], Diego Riofrío[1], Marta Berrocal-Lobo[2], Angélica de Antonio[1]
[1]Facultad de Informática, [2] ETSI Montes (CBGP)
Universidad Politécnica de Madrid
Spain
jramirez@upm.es, d.riofrio@alumnos.upm.es, m.berrocal@upm.es, angelica@fi.upm.es

Abstract— **This paper describes the preliminary results of an educational innovation project in which virtual world technologies have been used to build up a set of virtual laboratories oriented to engineering studies. Focusing on the architecture of a biotechnology laboratory created in this project, we describe how this architecture will also be adopted in other labs under construction, so that these labs can also take advantage of the same educational benefits. Instead of relying on a commercial software solution, these labs have been created with an open source software infrastructure named OpenSim. The limitations found in OpenSim and the solutions that have been provided to overcome these limitations are shown.**

Virtual worlds; virtual labs; non-immersive virtual environments; engineering education; opensim

I. INTRODUCTION

Non-immersive virtual worlds in education have shown to be very attractive to students due to their detailed graphics and social facilities. Expensive, dangerous, or time-consuming are some of the properties of a real laboratory that justify tackling its virtualization. One of the most popular platforms for the development of virtual labs is SecondLife [1][2]. In the specific field of biotechnology, it is remarkable the initiative of Leicester University with the SWIFT project (see http://www2.le.ac.uk/projects/swift). In this project, a biotechnology virtual lab was created in SecondLife. However, Secondlife has several limitations, such as price (you have to pay a fee for the land parcel in which you build your virtual lab), and technical constraints such as (1) impossibility to create more than a maximum number of objects in your parcel, or (2) impossibility to avoid undesirable visitors.

Other non-proprietary (open software) options exist, offering inexpensive and more flexible development infrastructure, such as Open Wonderland[3], Sirikata[4] or Open Cobalt[5], but the most extended alternative is OpenSim[6].

This work has been funded by the Department of Educational Innovation at Universidad Politécnica de Madrid (UPM) for the course 2010-2011 under project IE100210147.

Virtual worlds created with OpenSim rely on the creation of interactive objects by means of simple primitives named *prims* which can be joined to create complex objects, and they can have the intended behavior by means of scripts written in a programming language named LSL. Besides, this language allows the specification of interactive aspects such as object-to-object communications, as well as the communication between objects and virtual world inhabitants (students' avatars).

OpenSim platform represents a good trade-off between quality and development-cost in comparison with other technologies for virtual environments, specially, if we consider immersive virtual environments. For that reason and others, OpenSim can be considered a very interesting alternative when building virtual labs for educational purposes.

Previous works exploiting OpenSim as a software infrastructure to provide students with non-immersive virtual labs were focused on high school students [7], but the learning autonomy level was low, constraining lessons to teacher guided lessons or very simple autonomous learning based on reading information panels or interacting with learning-by-example machines without communication between these machines. Although advanced tools were developed in order to help teachers in their virtual lab teaching, for example, by providing them with monitoring tools to observe the student's attention level during a virtual lesson, the machines built in the virtual labs were simple in terms of interaction with the student and evaluation of the student activity.

These limitations were also reported in virtual laboratories created in SecondLife. For example, the work of Petrakou [8] concludes that, in their experience (languages teaching), there is a need to support asynchronous interactivity.

The work presented in this paper has focused on engineering education and was oriented to autonomous learning. Human professors play a secondary role because they are not involved directly during the learning process, but they

get reports automatically generated by our system in order to evaluate the activity of their students in the virtual labs. You can see videos of the labs at http://www.youtube.com/user/PEIAUPMVirtualLabs.

This paper is structured as follows: section II provides an overview of the biotechnology virtual laboratory, section III shows the dependency-driven guiding tutor proposed, in section IV we show technical details about the system's architecture, section V highlights other application fields for this system and, finally, section VI offers conclusions and lessons learnt.

II. OVERVIEW OF THE BIOTECHNOLOGY VIRTUAL LAB

In our biotechnology lab, the practice is organized into several levels in which a set of targets must be achieved in a specific order, inspired by online multi-user games with which a high number of students are familiar. In the practice the students are expected to obtain a tree whose genome (genetic material that defines an organism) has been modified in only one gene (fragment of the genome responsible for a concrete physiological function), increasing the levels at which this gene is expressed in the plant, by using genetic engineering techniques. In order to achieve the targets, the students can perform different kinds of actions such as mixing substances in a tube, applying some substances to small pieces of a plant, or allowing a plant to grow in places specifically prepared for that. The student will control the growth of the modified tree in axenic conditions at the laboratory following the process and acquiring skills on currently used biotechnology laboratory techniques.

After the completion of the practice the student is expected to conclude that the function of this modified gene is to codify or produce an antibiotic that leads to increased resistance of the tree towards certain diseases. The overall goal of the virtual lab is to facilitate the student the acquisition of skills in several basic plant genetic engineering techniques for genome modification, manipulation and study, as well as basic knowledge about plant *in vitro* growing.

The implemented practice takes students approximately 30 minutes to complete, whereas this same work would require almost two years in a real world laboratory due to the long waiting times associated to plant growing, a constraint that can be easily overcome in a virtual lab where the time scale can be manipulated as necessary. Moreover, the resources required for supporting the practice are too expensive to make it available to all the students of a related subject.

III. GUIDANCE AS AN ADDED VALUE

We consider that a virtual lab should support the execution of practical lessons in a way as faithful to reality as possible. Consequently, a virtual lab should be much more than a set of interactive objects. In the real world, the practical lessons performed in a lab use to be like a game, in which a set of targets must be achieved in a very specific order.

Accordingly, in our pedagogical approach, we aim to support one or more students carrying out a practical lesson by performing a sequence of steps or actions under the supervision of the system.

Guidance of the learning process is a very important aspect. When a student does not achieve a given target, there should be a mechanism to provide the student with hints and explanations about the reasons of his/her failures, or otherwise the student may get frustrated. For this purpose, we have created a Guiding Tutor software component that is capable of communicating with all the interactive objects involved in the learning environment, and that aims to provide a suitable guidance to the student. The guiding tutor is a key contribution to enhance the user's experience when compared to traditional non guided virtual labs. Figure 1 shows an example of a message provided by the guidance system.

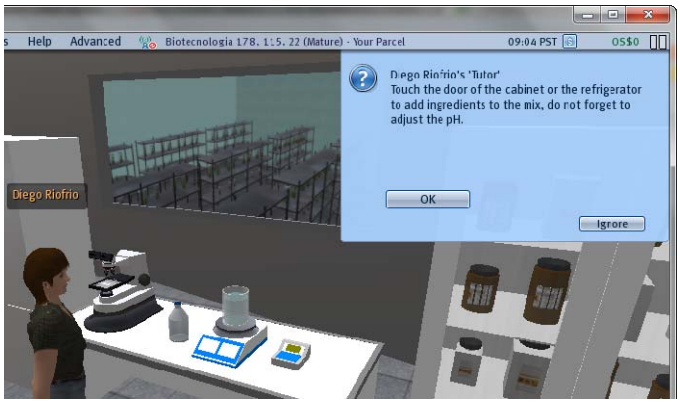

Figure 1. Example of guiding messages provided by the tutor system.

A. *Tutor guidance driven by dependencies*

In order for the Guiding Tutor to accomplish its mission, whenever the avatar of a student attempts to perform a relevant action over an object (touch, attach, drop, etc.), this object must inform of this action to the Guiding Tutor. If the action is right, the Guiding Tutor will give permission to the object to execute the action. Otherwise, the Guiding Tutor will register the student's error and will choose between giving or denying the permission to execute the action according to the tutoring strategy specified in design time. In some cases, also specified by the tutoring strategy, the Guiding Tutor will provide a text message to the student informing him/her of his/her error or giving him/her a hint.

Our tutoring guidance relies on dependencies defined among the steps or actions that the student has to perform in the practice. The next enumeration shows the different types of dependencies that have been defined.

1) *Previous Steps dependencies*

The practical lesson is structured as a step by step process, requiring a positive evaluation of some previous steps to be allowed to start the next one. This evaluation can take the form of an explicit test or a check of well-done activities. At the end of a given step, the tutor system may provide the student with a message, indicating a positive evaluation of the current step

and an invitation to proceed to the next step, or indicating a non-successful situation, because some required previous steps were not accomplished.

2) *Order dependencies*

In order to allow the student to perform some action, it may be required that some previous steps are executed in a given sequence, or in other cases, they can be completed in any order.

3) *Time and duration dependencies*

Some actions may require that a certain time has elapsed since a previous event. For example, in some cases, a machine can be used only if it was turned on some time ago. If you try to use that machine before that time elapsed, the tutor will indicate (immediately or at the end of the stage) that the machine is not available yet.

Also, it can also be specified that an action expires after a certain period of time.

4) *Action incompatibility (action context)*

Usually, an action is considered as correct if a previous set of actions are executed but, in some cases, an action is considered correct if a previous action (or actions) is (are) not executed. For example, once the action of turning on a light has been executed, the room will be illuminated until a turn-off action is executed. A given activity that depends on the execution of a previous set of actions can additionally depend on not turning off the light during the execution of these actions. Even if other dependencies were respected, if the light was turned off, the result will not be achieved.

IV. DETAILS ABOUT THE SOFTWARE ARCHITECTURE

Central to our architecture is the component that checks whether each student's action is correct with respect to the current state of the world and the provided plan for the practice. This central component will be called the Guiding Tutor. As it was previously mentioned, the behavior of the Guiding Tutor is driven by a set of dependencies that should be defined in the so called Tutoring Strategy, during the design of the virtual lab.

A. *Specification of the Tutoring Strategy*

The tutoring strategy is defined by means of an OpenSim notecard that is in the Guiding Tutor's inventory. This notecard is divided into lines, where each line corresponds to a certain student's action. For each action, among other issues, the tutoring strategy specifies the conditions that must be met so that this action can be executed. These conditions are expressed in terms of dependences with previously executed actions. So, if, for example, a student attempts to open a tap to fill a glass, and the student did not put the glass under the tap previously, the Guiding Tutor will not give the permission to execute the action "open the tap". The tutoring strategy also specifies the text message to inform the student of his/her error. When an action depends on more than one previously executed action, a different error message may be specified in the tutoring strategy for each dependency.

As was mentioned in section III, when defining the dependencies with previous actions, we may face cases where these previous actions must be executed in a certain sequence, or cases where they can be executed in any order. In order to illustrate this, we will employ the following example. Let us suppose that the student, as part of the practical lesson, has to prepare a mixture of several substances, in particular, essential nutrients (Murashige & Skoog), sacarose and agarose. These substances must be added to a glass with water as long as this glass is over a running shaker. After adding the essential nutrients and the sacarose, the student must adjust the ph of the mix by using a ph-meter, and then he/she must add the agarose. Finally, he/she must turn off the shaker. In this practice, it does not matter if the student adds first the essential nutrients and then the sacarose, or vice versa. This plan or lab protocol is outlined in figure 2. As can be deduced from this plan, there are two right sequences of actions for carrying out the whole plan.

Moreover, apart from being aware of the plan for the practical lesson, the Guiding Tutor needs to know how to deal with the situations where the student may fail. We will see how the treatment for these failure cases is specified as part of the tutoring strategy by focusing on the treatment of the last action "turn off the shaker".

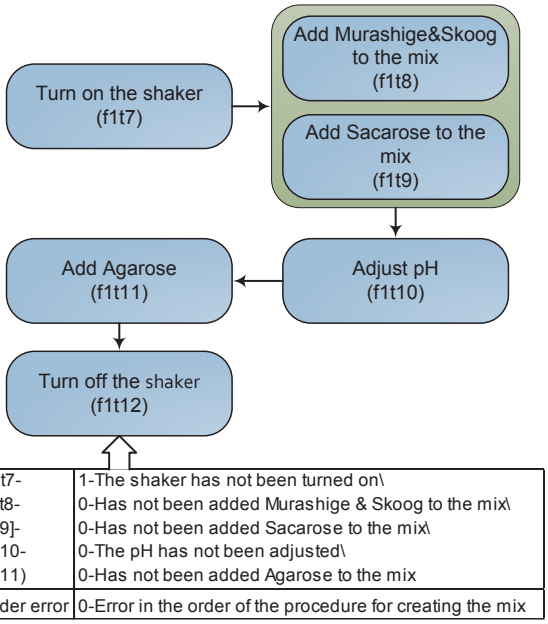

Figure 2. Example of a plan in the biotechnology virtual laboratory.

When deciding how to deal with each of the different failure cases associated with the action "turn off the shaker", we have considered pedagogic aspects suggested by a human professor. For example, it may be preferable not to reveal to the student certain errors just after he/she committed them, because it is better that the student sees the final consequences of his/her wrong actions. In this example, the final consequence will be a plant with visible growing problems. Taking into account this, the Guiding Tutor will not inform the student if he/she forgot to add some of the substances mentioned before, but it will actually register those errors for providing a subsequent feedback to both the student and his/her professor. However, the Guiding Tutor will actually inform the student that he/she did not turn on the shaker These decision rules for

the last action are outlined in figure 2. The prefix '1-' represents that whenever the action f1t7 is not done, the message "The shaker has not been turned on" will be shown to the student. On the other hand, if any of the other actions is not done, its associated message (with prefix '0-') will not be shown to the student, but it will be registered by the Guiding Tutor.

In figure 2 there is also an error message defined for the case in which the student has indeed carried out all the required actions, but not in the specified order.

It is also worth mentioning that all the actions except for "turn on the shaker" have an incompatible action: the action "turn off the shaker". If the shaker is turned off at any moment before all the previous actions have been completed, the mixture will not be properly shaken and the final result will not be satisfactory.

Once the student performs an action correctly, the tutoring strategy will specify whether the system must indicate the next action to the student or not. This mechanism will be specially interesting for those situations where the human professor wants to help the student or, on the contrary, prefers the student to find out by himself/herself the next step of the practice. Moreover, thanks to this mechanism, it will be easy to adapt the same virtual lab to students with different knowledge of the matter (for example, undergraduate and graduate students).

B. *Walkthrough the Software Architecture*

In this section we will show a dynamic view of the virtual lab operation. More precisely, we will focus on the exchange of messages between the Guiding Tutor and other objects that need to validate the actions attempts. For illustrating this exchange of messages, we will take as scenario of study the previous example where a student through his/her avatar has to prepare a mixture of several substances.

In figure 3, we can see the 25 messages that are exchanged, mainly between the Guiding Tutor and other objects. This scenario begins when the avatar of a student touches the shaker to switch it on. Next, the shaker sends a message to the Guiding Tutor to validate the attempt of switching on the shaker. To this message, the Guiding Tutor can reply giving (OK) or denying (FAIL) the permission to switch on the shaker. If the permission is given, the shaker sends a message to the glass to shake the mixture inside the glass. Then, in order to add the essential nutrients (M&S) to the glass, the avatar touches the chemical cabinet to open its doors, and a dialog is created and shown to the user. In this dialog there is a button for each substance available in the cabinet. As soon as a button is pushed (let us suppose that the button associated with the essential nutrients), the dialog informs the cabinet of this, and then the cabinet asks the Guiding Tutor to validate the action "take M&S". Subsequently, similar messages are exchanged for performing and validating the actions "take sacarose", "adjust pH" and "take agarose". In the case of the action "turn off the shaker", as it was explained in the previous subsection, it may happen that the Guiding Tutor does not validate the action, but the execution of the action is nevertheless allowed. In that case, the Guiding Tutor replies with a FAIL_BUT_GO message.

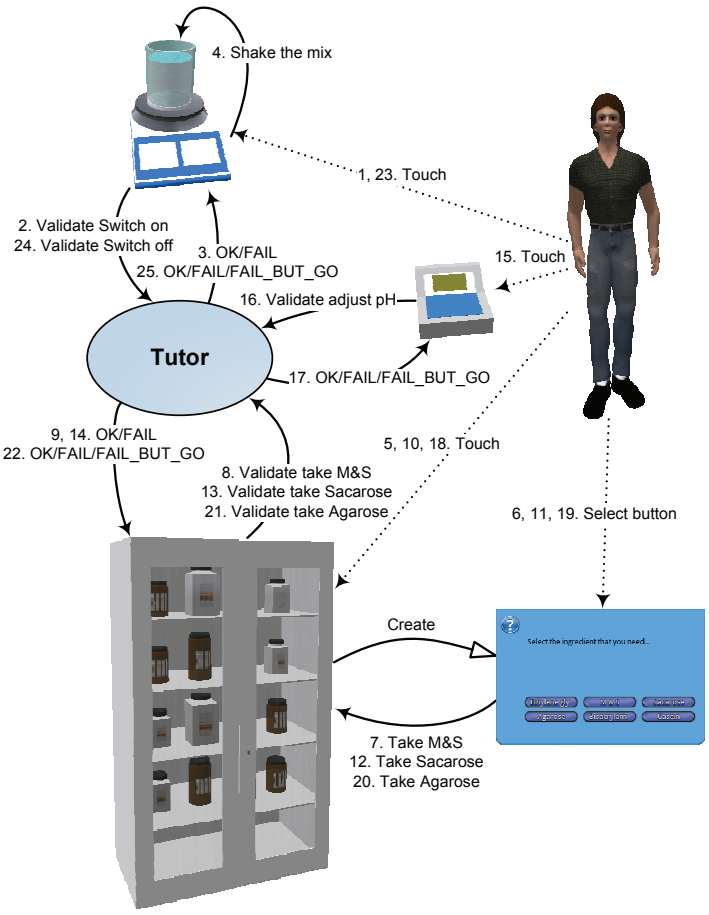

Figure 3. Messages exchange between the guiding tutor and learning objects.

V. OTHER APPLICATIONS OF THE GUIDING TUTOR

The Guiding Tutor was designed so that it can be easily configured to provide different tutoring strategies in the biotechnology virtual lab. Moreover, even if the Guiding Tutor component has only been employed in the biotechnology lab so far, from the beginning this element was conceived as a general-purpose component that may be reused in different virtual labs. In this section, we will explain briefly how the Guiding Tutor will provide an educational added value to other virtual labs currently under construction.

- **Electronics virtual lab**: in this lab the student has to connect correctly a circuit to an oscilloscope, a multimeter, a power supply and a wave generator by means of some cables, and then he/she must observe the graphs and the values in the displays. Here, the Guiding Tutor should validate the sequence of actions required to set up the connections, warning about wrong connections or giving hints related to the next actions.
- **Irrigation of a plantation virtual lab**: here the student has to program the calendar of irrigation for a plantation, keeping in mind the weather forecast and

the amount of available water. In this context, the Guiding Tutor may should about wrong irrigation programming, considering, for example, that it is not necessary to irrigate when it is raining. It may also warn when it is urgent to irrigate the plantation, because the ground is too dry.

We are also planning the application of the Guiding Tutor to already existing virtual labs that will be ported to OpenSim:

- **Nuclear power plant virtual lab**: the students play the role of future maintenance operators that need to learn protocols related to the maintenance of the nuclear power station. In this case, the role of the Guiding Tutor would be very similar to the one in the biotechnology virtual lab because of both virtual labs are intended for procedural training.
- **Chemistry virtual lab**: the student learns to perform different chemical experiments, some of them involving the manipulation of dangerous materials, such as sulfuric acid. The Guiding Tutor will supervise all the procedure, making sure that the steps are taken in the proper sequence and to avoid accidents.

Additionally, we are in the process of extending the capabilities of the Guiding Tutor in order to allow the students to ask for help when they need to locate a certain object or place in the virtual space, and to be able to track the movements of the students along the virtual environment and to evaluate the extent to which they are following optimal routes.

VI. CONCLUSIONS

We have designed a mechanism to develop guided virtual labs by using a free infrastructure for the development of virtual worlds, namely OpenSim. Our approach provides an inexpensive and flexible solution that can be easily configured to different practices and labs. The tutoring strategy is based on the definition of dependencies between actions, and it is very easily defined in an OpenSim notecard, allowing even a human professor to specify it without any programming. The Guiding Tutor is a reusable software component that can be easily integrated into new virtual labs, providing the student with the necessary supervision and guidance during their learning process. The human professor is involved by providing timely information about the student's performance. In fact, the evaluations coming from our virtual labs can be combined with other grades in Moodle, so that the professor can smoothly integrate virtual labs with other learning activities.

During our experience developing virtual labs with OpenSim we learnt that a high level of realism and a high educational value can be obtained by adopting our extension to this open code (royalty free) software system.

ACKNOWLEDGMENT

Authors thank UPM Educational Innovation Program for their support. M. R. thanks Spanish national (MICINN) and local institutions (CAM) for their support under projects TIN2011-24139, S2009/TIC-1650 and TIN2008-02081. A. de Antonio thanks MICINN for their support under project TIN2009-14659-C03-02.

REFERENCES

[1] M. N. K. Boulos, L. Hetherington and S. Wheeler, "Second Life: An Overview of The Potential of 3-D Virtual Worlds In Medical and Health Education," Health Information and Libraries Journal, 24(4), 2007, pp. 233–245.

[2] Linden Lab, "Second Life Education: The Virtual Learning Advantage", technical report, available at http://lecs-static-secondlife-com.s3.amazonaws.com/work/SL-Edu-Brochure-010411.pdf, 2011.

[3] J. Kaplan, N. Yankelovich, "Open Wonderland: An Extensible Virtual World Architecture", IEEE Internet Computing, 15(5), 2011, pp. 38-45.

[4] D. Horn, E. Cheslack-Postava, B. Mistree, T. Azim, J. Terrace, M. Freedman and P. Levis, "To Infinity and Not Beyond: Scaling Communication in Virtual Worlds with Meru", Technical Report CSTR 2010-01, Dept. of Computer Science, Stanford University, 2010.

[5] See http://www.opencobalt.org

[6] See http://www.opensimulator.org

[7] M. Rico, G. Martínez-Muñoz, X. Alaman, D. Camacho and E. Pulido, "A Programming Experience of High School Students in a Virtual World Platform," International Journal of Engineering Education, vol. 27, 2011, pp.52–60.

[8] A. Petrakou, "Interacting through avatars: Virtual worlds as a context for online education", Computers & Education, vol. 54, 2010, pp. 1020–1027.