Integration of simulators in virtual 3D computer science classroom

Irina Branovic, Ranko Popovic
IEEE Member, Serbia

Nenad Jovanovic
University of Kosovska Mitrovica, Serbia

Roberto Giorgi
Department of Information Engineering, University of Siena, Italy

Bosko Nikolic, Miodrag Zivkovic
School of Electrical Engineering, University of Belgrade, Serbia

Keywords - 3D environments; simulation; virtual worlds; computer science education; modular systems

Abstract - **In this paper we describe a virtual 3D lab created using Open Wonderland to enable simultaneous execution of different computer science simulator modules in a collaborative, immersive workspace. Simulators were divided into functional modules that can be reused and combined for different teaching topics based on statistics of their usage. We discuss some important issues in our platform setup, describe our learning environment and logging module we developed in order to track system usage. Preliminary usability testing confirms the efficiency of this approach.**

I. INTRODUCTION

Providing suitable learning environments in the real world is sometimes very difficult; virtual environments have the potential to provide immersive learning spaces and open up new and previously unimagined possibilities. 3D environments overcome the need for learners to be physically present and provide resources that would otherwise be expensive or inaccessible. By shifting the context of learning into customized environments on the web, a new world of teaching and learning can be explored.

A common point shared by simulation tools and 3D environments is that they have been widely used to study a variety of problems which are too costly or impractical to do in real life. This is especially true in engineering education, where it is very important for students to be able to observe, explore, and manipulate device characteristics and behavior.

Simulators are usually developed by experts who have an in-depth understanding of the topic being modeled, but are designed to be executed independently, and often created in different programming languages. The challenge is to facilitate the process of executing independent simulators together in a common environment where students can explore different alternatives and conduct low cost experiments. Also, integrated concurrent and parallel execution of simulators is essential to understanding the effect of the same phenomena and to conduct what-if analysis, thereby improving students' understanding of the subject taught.

The work presented here has been supported by the Serbian Ministry of Education and Science (project grant no. III44006).

In our previous work [1] we described initial efforts in designing a lab for introductory computer science courses. Following this work, here we describe a continuing process of extending a 3D computer science lab with pre-existing simulators which range in topics from digital logic design, through pipeline and processor cache simulation, to computer network simulation.

We have several objectives in this paper. First, we describe steps taken for integrating diverse simulators divided into modules within 3D classroom. Next, we describe our learning environment through a case study which illustrates how it facilitates the use of different functional modules to gain a "big picture" of the problem. Further, we describe a log module we developed for tracking students' activities and creating statistics of system usage. Along the way, we discuss important issues of platform setup which can be helpful to others who deploy a virtual world for their own purposes. Finally, we report on preliminary results of system usage and present our plans for future work.

II. 3D AND SIMULATION

To setup a classroom, we used Open Wonderland [2], an extensible open source toolkit for creating collaborative 3D virtual and mixed-reality worlds. It supports immersive audio, desktop application sharing, and integration with external data sources. Paper [3] outlines some of the key advantages and drawbacks of choosing Open Wonderland as a virtual world platform for teaching. Especially useful for our purpose is its platform independence, due to its Java codebase. Additionally, it is relatively easy to extend the platform by developing new modules. In comparison to commercial alternatives such as Second Life, Wonderland enables greater control over resource access, privacy and security [4]. Choosing an open source platform reduces initial costs for setting up the virtual classroom, but requires hosting dedicated server and programming efforts to further develop the software.

We integrated a number of simulators in active use in education into a common virtual classroom. Instructors can freely choose among different simulators to use for teaching: SDLDS [6] for digital logic design, JCacheSim [7] for processor cache, WebMIPS [8] for pipeline simulation, general computer system simulation SIMAS [9], or eWiSENS [10] for wireless sensor networks. Simulators can be added to a scene as independent objects in Wonderland, but more important, teachers are also offered separate functional modules extracted from each simulator in order to create a unique toolset suitable for the problem being taught. Conditions that every simulator must meet to be integrated in our classroom are that it is open-source, free for educational use, and programmed in Java programming language with modular approach in mind. Two are the principal reasons for that: first, the chosen virtual world tool, Open Wonderland, requires modest programming effort to integrate any Java software as a module. Importing simulators does not require code rewrite; instead, individual simulators (and their modules) are able to maintain their autonomy, thereby avoiding code rewrite. Second, Java's known native support for internationalization (I18n) makes it very easy to create versions of the same classroom in different languages.

Knowing from [4] and our previous experiments that Open Wonderland ideally requires a 'modern game hardware' client with powerful graphics capability, we were able to avoid all performance and usability related issues in clients running with different graphics configurations.
Java-based simulators are integrated using a Wonderland module template. Modules are pieces of Java code that envelope a feature or a functionality that can be plugged in into a core Wonderland project; every module has its 3D representation and functionalities that can be added to it.

Further, we identified the functionalities of each available simulator that can be used separately and organized them into modules that can be included in the scene as objects.
Wonderland also offers VNC Viewer module that allows displaying a remote desktop and interacting with it from within environment. This module can potentially be used to integrate any software running on a remote machine into the environment; however, having the source code of simulators available offers notable benefits, such as the possibility to modularize it for reuse, trace the way simulator is used internally, or implementing I18n.

III MODULARIZATION OF SIMULATORS

Previous work dedicated to integrating simulators such as [5] mostly concentrated on providing middleware solutions for creating common environments for execution of simulators, to avoid code rewrite. Our approach is based on the idea of modularization, which requires a modest programming effort to conform to Wonderland module template; to implement it, we categorized common functionalities of existing simulators based on envisaged teaching needs.

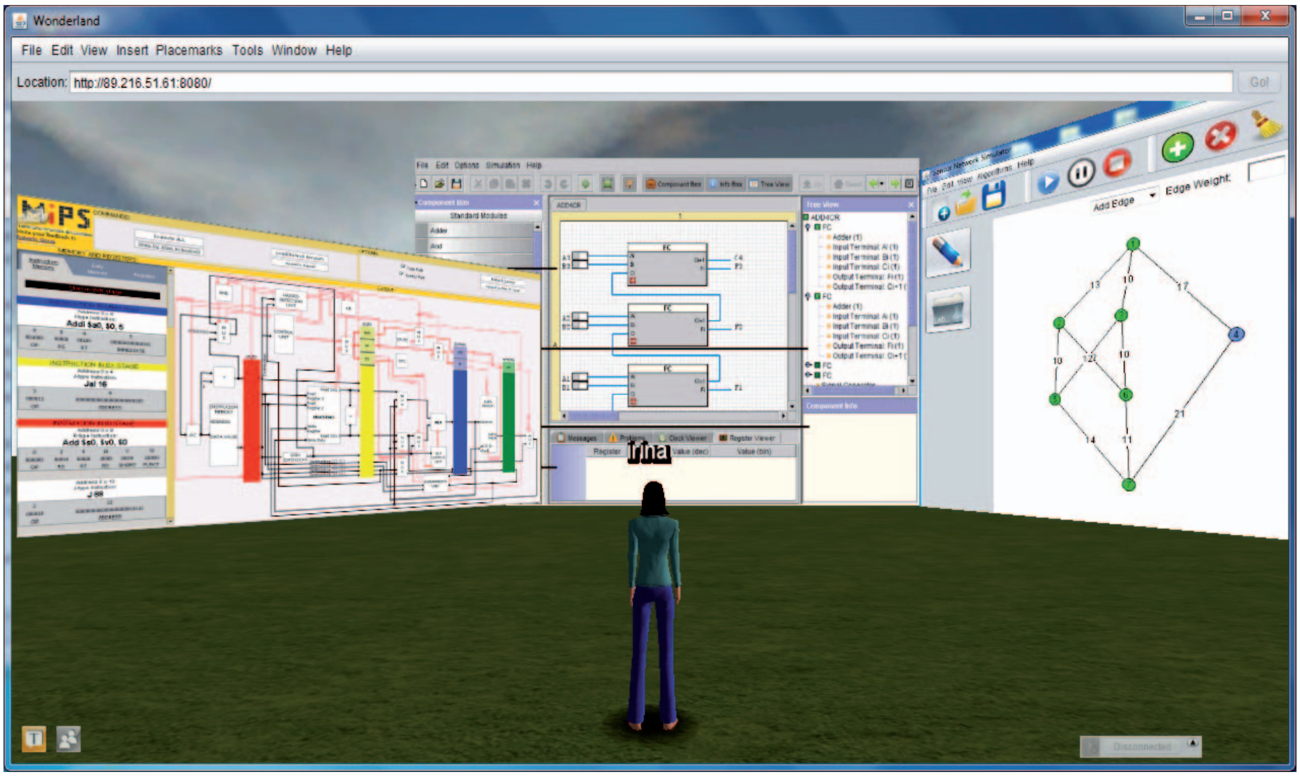

Wonderland
File Edit View Insert Placemarks Tools Window Help
Location: http://89.216.51.61:8080/
Go!
MIPS
Add Edge
Edge Weight
Trina

Figure 1. A scene from virtual classroom involving three complete simulators.

Specifically, SDLDS simulator [6] is divided into following modules:

- designing switching circuits
- simulation of switching circuit
- verification of the switching circuit implementation

JCachesim [7] is modularized into:

- learning assembly coding
- simulating cache memory
- simulating main memory

WebMIPS [8] is divided into modules for:

- learning assembly coding
- simulating MIPS pipeline
- simulating the content of processor registers

SIMAS simulator [9] is divided into following modules:

- working with numerical systems
- simulating switching circuits
- simulating the complete computer system
- simulating algorithms and data structures
- routing
- IP networking
- process tracking
- reliability and fault tolerance in wireless networks

eWiSENS [10] is modularized into:

- developing wireless sensor network architecture
- routing
- WSN security
- sensor energy efficiency
- spectral graph theory

Interdisciplinary teaching subjects can especially benefit from our setup; some courses suitable for our environment are computer architecture, operating systems, networking and communication, parallel and distributed systems, intelligent systems. For example, when teaching RISC processor architecture, a teacher can select modules from SIMAS, MIPS, JCacheSim, DSLDS, or eWiSENS. Similarly, when teaching computer memory related topic, modules can be selected from either SIMAS or JCacheSim; when teaching routing, we might use appropriate SIMAS or eWiSENS module. To help instructor decide which among completely or partially overlapping modules to put in a scene, we developed a logging system which we describe next.

IV LOGGING STUDENTS' ACTIVITIES

Besides offering all available simulators and corresponding modules in a common environment, our system is able to follow activities of students within the classroom and to suggest appropriate modifications of the scene.

The role of an instructor in 3D classroom is very important and dominant, especially in providing teaching materials, discussion topics, and exercises. However, students are free to join the classroom anytime, not only in scheduled sessions; this especially benefits students who prefer self-learning over group learning. It is essential that teachers are aware of the options available to them to follow and monitor students as effectively as possible since this is of enormous pedagogical importance.

Although our logging system tracks individual students and their sessions, our initial goal here was not to follow individual students' learning habits, but to discover how users of our system tend to spend time within the environment. Armed with that information, teachers can intervene preemptively and undertake steps such as enriching educational experience by adding or removing elements in the scene, promoting active and collaborative learning, involving a particular student to interact more with others.

Logging is added as a separate module into Wonderland Web administration screen (Figure 2). During the development of logging system we were able to leverage existing object hierarchy of the scene and our composite-component model of arranging scene components, as described in [1]. The fact that Wonderland already has the ability of recording active users and their coordinates in the 3D space was extremely helpful.

Logging is implemented as related to a particular scene layout created by instructor for a specific teaching subject. Initial testing was organized in such a way that particular scene layout was tracked for a certain time period (a month or two), log file was analyzed, and based on conclusions scene layout is modified. Since Wonderland is able to create snapshots, i.e. to remember server states in different points in time, returning to previous scene layout and the corresponding log file is easy.

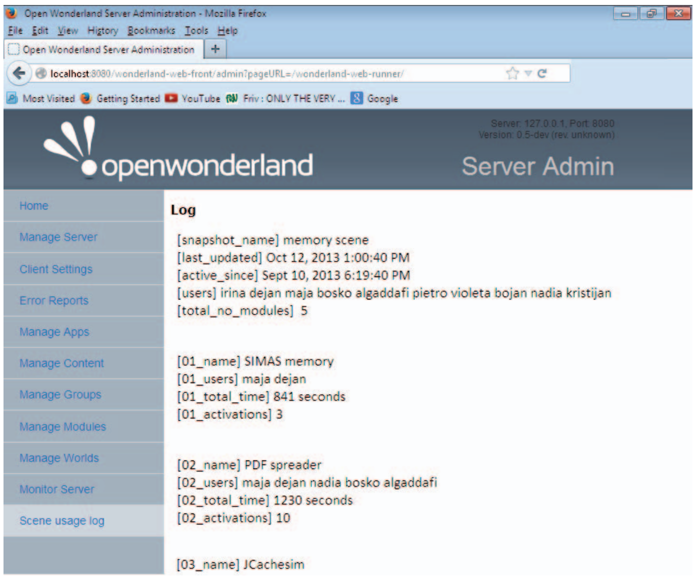
Typical testing classroom sessions involved a group of ten students and a teacher connecting to the Wonderland server, navigating the virtual space, and interacting with the simulation tools and other objects present in the scene. Log data gathered through the sequence of sessions provided the raw information for analysis; students' interaction while carrying out the tasks (i.e., screen and audio recordings) is also observed and analyzed. The purpose of the analysis was to determine the factors that influence the activeness of the students and to determine usefulness of components added to the scene. Log system is able to help teachers in two ways:

- to identify components in a scene which are most useful (i.e. those with largest number of interactions and/or time spent interacting with)
- to identify the preferred among overlapping modules, i.e. choosing the component preferred by students for a specific teaching subject.

V CONCLUSION AND FUTURE WORK

Once the ideal module and component combination is established, scene layout should remain mostly static, and attention should be given to considering each student's learning preferences and skills. Most models and theories of learning presented in the literature concentrate on developing learning types and categorizing them into different dimensions [11], such as:

- *active/reflective learners*: learn by direct interaction with the material; prefer group communication
- *sensing/intuitive*: detail-oriented and practical with a preference for concrete facts and real-world applications
- *visual/verbal*: better able to remember images they have seen, and/or written or spoken words
- *sequential/global*: prefer learning linearly, with logical steps, or by fitting pieces together into a big picture.


Open Wonderland Server Administration - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Open Wonderland Server Administration
localhost:8080/wonderland-web-front/admin?pageURL=/wonderland-web-runner/
Most Visited Getting Started YouTube Friv : ONLY THE VERY ... Google
openwonderland
Server: 127.0.0.1, Port: 8080
Version: 0.5-dev (rev. unknown)
Server Admin
Home
Manage Server
Client Settings
Error Reports
Manage Apps
Manage Content
Manage Groups
Manage Modules
Manage Worlds
Monitor Server
Scene usage log
Log

```
[snapshot_name] memory scene
[last_updated] Oct 12, 2013 1:00:40 PM
[active_since] Sept 10, 2013 6:19:40 PM
[users] irina dejan maja bosko algaddafi pietro violeta bojan nadia kristijan
[total_no_modules]  5


[01_name] SIMAS memory
[01_users] maja dejan
[01_total_time] 841 seconds
[01_activations] 3


[02_name] PDF spreader
[02_users] maja dejan nadia bosko algaddafi
[02_total_time] 1230 seconds
[02_activations] 10


[03_name] JCachesim
```

Figure 2. Typical content of a log file shows time and space usage of classroom components (simulator modules) for separate sessions and individual participants.

The next step in developing logging module is to enable tracking statistical data on participant behavior and preferences and offer a personalized version of the classroom to each student, based on his individual learning styles.

Our system is currently used as teaching aid for different courses (digital logic design, computer architecture, operating systems, and computer networks). Now that we have overcome most of the usability issues in deploying our tools, our goal is to further refine the logging tools and integrate more automatic activity assessment prior to further user evaluation. The evaluation module of the system is particularly important and we plan to further develop it with a data warehouse for storing information about using the classroom and particular modules at a very detailed level.

REFERENCES

[1] I. Branovic, D. Markovic, V. Tomasevic, D. Zivkovic, R. Popovic, "Development of modular virtual lab for introductory computing courses," Proceedings of IEEE EDUCON Conference, 2013, pp. 1027–1031.

[2] Open Wonderland Community (2013) [Online]. Available: http://openwonderland.org/

[3] M. Gardner, J. Scott, B. Horan, "Reflections on the use of Project Wonderland as a mixed-reality environment for teaching and learning", Proceedings of Researching Learning in Virtual Environments. The Open University, UK, 2008, pp. 130–141.

[4] D. Parsons, R. Stockdale, "Cloud as Context: Virtual World Learning with Open Wonderland", *Learning*, 2009, pp. 123-130.

[5] Jalali L., Sharad M, Nalini VV, "Middleware solutions for integrated simulation environments", Proceedings of the 7th Middleware Doctoral Symposium, pp. 2-7, ACM, 2010.

[6] Stanisavljevic, V. Pavlovic, B. Nikolic, J. Djordjevic, *SDLDS—System for Digital Logic Design and Simulation*, IEEE Transactions on Education, Vol. 56, No. 2, 2013.

[7] I. Branovic, R. Giorgi, A. Prete, "Web-based training on computer architecture: The case for JCachesim", Proceedings of WCAE Workshop, 2002.

[8] I. Branovic, R. Giorgi, E. Martinelli, "WebMIPS: a new web-based MIPS simulation environment for computer architecture education", Proceedings of workshop on Computer architecture education, 2004.

[9] N. Jovanovic, D. Markovic, D. Zivkovic, R. Popovic, *SIMAS: A Web-Based Computer System Simulator*, International Journal of Engineering Education, Vol. 26, No. 3, 2010, pp. 1–8.

[10] M. Zivkovic, B. Nikolic, R. Popovic, "eWISENS – Educational Wireless Sensor Network Simulator," unpublished.

[11] R. M. Felder, L. Silverman, *Learning and Teaching Styles in Engineering Education,* Engineering Education Journal, vol. 78, 2008, pp. 674—681.