Proceedings of the 9th IFAC Symposium Advances in Control Education
The International Federation of Automatic Control
Nizhny Novgorod, Russia, June 19-21, 2012

Synchronous collaboration between auto-generated WebGL applications and 3D virtual laboratories created with Easy Java Simulations

Carlos A. Jara* Francisco A. Candelas* Fernando Torres* Christophe Salzmann Denis Gillet****
Francisco Esquembre* Sebastián Dormido******

**University of Alicante, Carretera de San Vicente del Raspeig, s/n, PO Box 03690, Alicante Spain (e-mail: {carlos.jara, francisco.candelas, fernando.torres}@ua.es).*
***École Polytechnique Fédérale de Lausanne, CH-1015 Lausanne Switzerland (email: {christophe.salzmann, denis.gillet}@epfl.ch)*
****University of Murcia, Campus del Espinardo, PO Box 30003, Murcia Spain (e-mail: fem@um.es)*
*****National Distance University (UNED), C/ Bravo Murillo, nº28, PO Box 28040, Madrid Spain (e-mail: sdormido@dia.uned.es)*

Abstract: This paper presents a new collaborative e-learning system based on a real-time synchronized communication among virtual laboratories. This original approach provides a new tool which integrates virtual laboratories inside a synchronous collaborative e-learning framework. This system is based on the automatic generation of WebGL simulations from 3D Easy Java Simulations' applets. These WebGL simulations created are synchronized through the Internet in order to integrate them in an on-line collaborative environment. In this way, several students can attend in a virtual class using only a communication device using any web-browser with WebGL enabled. The paper also describes the software architecture in which is based on this new education tool.

Keywords: collaborative environment, distance teaching, Java 3D, virtual laboratories, WebGL

1. INTRODUCTION

Computer-Supported Collaborative Learning (CSCL) is an approach based on the constructivist and collaborative learning theories, which are focused on social inter-dependence and maintain that students consolidate their learning by teaching one another (Alavi, 1994). First CSCL environments were created for using technology as a mediation tool within collaborative learning methods of instruction (Koschmann, 1994). Currently, thanks to the great evolution of network technologies, CSCL is being integrating into Web-based platforms. Thus, education process can get out of the traditional classroom.

Considering the moment when the student-teacher interaction takes place, CSCL environments can be classified into asynchronous and synchronous (Bafoustou and Mentzas, 2002). The first one allows data exchange in flexible time-tables and remote access in an asynchronous way. However, this can cause feelings of isolation in the student, and hence reduces his/her motivation (Kamel, Taylor and Breton, 2005), because students do not receive instant feedback from their questions. In contrast, synchronous environments enable e-learning in a similar way to the traditional classroom, sharing experiences in real-time like face-to-face interaction.

However, current CSCL synchronous environments are mainly focused in the learning of theoretical lessons. When web-based tools are required for the distance education of practical concepts, especially in engineering, the best option is the virtual laboratories. By means of virtual labs, students can learn through the Internet in a practical way and thus become aware of physical phenomena that are difficult to explain from just a theoretical point of view (Dormido, 2004). Their interactivity encourages students to play a more active role in the e-learning process and provides a realistic hands-on experience (Dormido et al., 2005a). Nevertheless, the majority of the virtual labs included in Web-learning environments are designed to be used individually, and they do not allow work group. The true integration of virtual labs inside collaborative learning environments can be seen in eMersion (Gillet, Nguyen and Rekik, 2005) and Automatl@bs (Vargas et al., 2011). These Web-based platforms contain a series of applications whereby students can experiment and share results among other students or with teachers. However, the collaboration in these CSCL environments is in an asynchronous way.

The work presented here develops the combination of the virtual laboratories with a synchronous CSCL, merging the features and advantages of both paradigms. A first approach based on this idea was proposed earlier (Jara et al., 2009), but it supposed a certain complexity of implementation, mainly due to the communication protocols used. The novel solution presented in this paper makes use of new WebGL (Web-based Graphics Library) technology to greatly simplify the process of launching an online collaborative class. WebGL is based on a software library that extends the JavaScript programming language to allow it to generate interactive 3D graphics within any compatible web browser. WebGL also extends HTML with advanced computer graphics without the use of plug-ins. WebGL is managed by the non-profit Khronos Group, and its specification 1.0 was released on March 3, 2011 (WebGL - OpenGL ES 2.0 for the Web, 2011). The HTML pages generated with this system can be integrated in a Personal Learning environment in which collaborative learning services are integrated (Gillet, 2010; Salzmann and Gillet, 2011).

The remainder of this paper is organized as follows: Section 2 explains the collaborative system, detailing aspects such as components, communication framework and software architecture. Next, the automatic generation of WebGL simulations from 3D Ejs' applications is described. Afterwards, a complete example of a virtual collaborative class' generation will be shown in Section 4. Finally, some important conclusions are presented in Section 5.

2. SYSTEM OVERVIEW

In this section, a detailed system description is explained. Firstly, the main components which make up the collaborative system are described. Next, the communication tools used in the real-time collaboration and how the simulations are synchronized are explained. Afterwards, the software architecture of the components is presented.

2.1. Components of the collaborative environment

The main objective of the collaborative system developed is to offer a simulation (which is generated automatically from a 3D Ejs' applet) that can be easily visualized in real-time by different members of a virtual class (i.e. students) with only a web-browser. Therefore, this e-learning system has two main components: a main applet which is created with Ejs and some students' WebGL simulations embedded in a HTML page. The main applet is a 3D virtual laboratory which manages the virtual class controlling the simulation evolution in real-time. WebGL simulations cannot interact with the shared simulation and they only are able to show what the main applet is doing on the virtual laboratory.

2.2. Communication framework

The communication framework of the collaborative system presented is based on JavaScript Web-Sockets (The Web-Sockets API, 2011), a new and powerful tool based on the TCP protocol. A Web-Socket defines a full-duplex single socket connection over which messages can be sent between client and server, both embedded in JavaScript code on a HTML page. This technology greatly simplifies the complexity around bi-directional web communication.

WebGL simulations are connected around the main applet in a peer-to-peer (P2P) centralized overlay network using Web-sockets (Fig. 1). The main applet contains a communication module which manages the synchronization of all the WebGL simulations connected in the virtual class.

The system is focused in a server-less architecture. This communication method provides several advantages. First, it avoids the delay caused by the web server processing in the data flow. Second, users do not have to install any centralized server program in the web server because the communication engine is embedded in the HTML simulations. And third, the number of network connections is lower than in a system with a web server in the architecture.

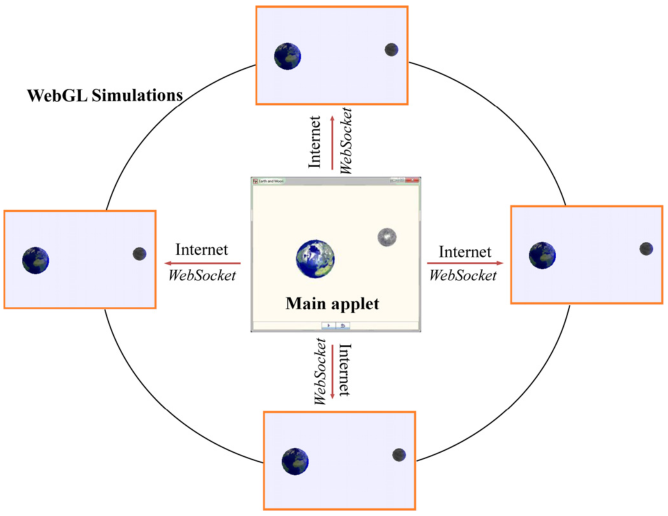


Fig. 1.Centralized network between the main applet and the WebGL simulations

To synchronize the shared virtual laboratory (3D applet and students' WebGL simulations), the applications must be in the same state for the model all the time. As it will be commented in Section 3, WebGL simulations generated only are based on the view part of Ejs' applications. Thus, WebGL simulations do not have model and they are updated with the interface data provided by the Ejs' applet. This simplifies the communication framework and the WebGL simulations.

The update messages sent from the 3D application to the WebGL simulations only transmit the state of the view or interface elements, as for example positions and orientations of the 3D objects. Same state means that simulations must be in the same step. Therefore, for each simulation step performed in the main applet, all the needed information to refresh all the WebGL simulation is sent via the Web-Socket established. In order to achieve a synchronization in real-time among all the simulations connected in the virtual session, after sending a new packet of information for each simulation step, the main applet pauses the simulation evolution and waits for the response from WebGL simulations to verify that they have been updated. After receiving this response, the main applet continues with the system evolution.

2.3. Software architecture of the communication engine

The communication system explained in the previous subsection is embedded both in the main applet and in the WebGL simulations. This subsection describes the main applet and the WebGL simulation communication engine which are the components of the software layer of the synchronized collaborative environment.

2.3.1. Main applet architecture

The main applet is thread system that manages the synchronization among different WebGL simulations connected to the virtual class. The main parts which compose its communication system are shown in Fig. 2.

At the beginning of the virtual session, the teacher applet executes an instance of a Web-Socket Server object which attends new Web-Sockets client requests from the WebGL simulations in a fixed port (n_m). For each simulation connected, an array of Web-Socket connections is updated.

To synchronize the applications of the shared virtual laboratory, the main applet retransmits update messages to WebGL simulations. As mentioned before, these commands basically are interface data, and positions and orientations of the 3D objects which compose the virtual environment. In addition, as commented before, there is a receiver module which is in charge of receiving response confirmations of WebGL simulations to verify that they have been updated.

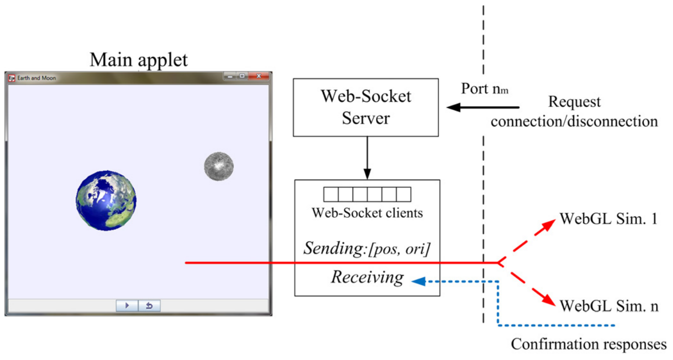


Fig. 2. Main applet software architecture

2.3.2. WebGL simulation architecture

WebGL simulation's software architecture is shown in Fig. 3. This is is embedded in the HTML page downloaded by the user. It is simpler than the main applet's architecture.

There is a Web-Socket client which connects with the main applet. The WebGL simulation receives the interface data of the main applet and it is able to perform the suitable changes in the application. In addition, for each change performed, a confirmation to the main applet is sent in order to synchronize all the applications of the virtual class (see in Fig. 3 the text "confirmation responses").

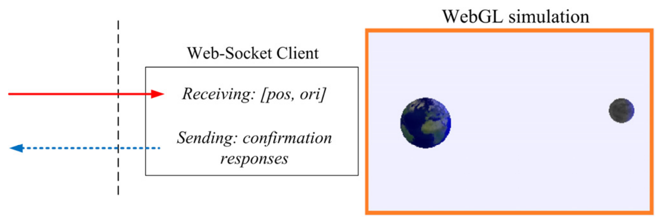


Fig. 3. WebGL simulation software architecture

3. WEBGL GENERATION

All the applications developed with Ejs have a software core divided in two main parts: model and view. The model is in charge of computing the value of the variables which describe the system, and the view or interface is based on a set of standard Java Swing, Java 2D and Java 3D components (in this paper, authors only have considered views created with 3D components). WebGL simulations generated are only based on the view part of Ejs' application. This way, WebGL simulations do not have any model engine and they are only updated with the interface data provided by the Ejs' applet which is sent via the Web-Socket communication.

Fig. 4 shows how WebGL simulations are created from Ejs's applications. Initially, the main applet sends via the HTTP protocol some information about the 3D environment of the applet, most of them in regard with the view of the simulation: 3D objects, lights, camera features, initial position and orientation of the virtual objects, etc. This information is passed as POST parameters to a PHP module which generates a HTML web page in the web server with the WebGL simulation embedded (file *Simulation.js*). This JavaScript simulation is created from a WebGL library based on the same view elements of Ejs.

4. EXPERIMENTAL EXAMPLE

In this Section, the process to create a virtual collaborative class from an Ejs' application is explained. The generation of the WebGL simulations has been integrated in the options of a new beta version of Ejs. Basically, the process is composed of two points:

- The generation of a Java applet using the new 3D framework of Ejs (Jara et al., 2011).
- The automatic generation of a URL address in a public web server, where students or users can download their WebGL collaborative simulations.

This section will be explained using a specific simulation, a 3D Futura pendulum. Authors do not deem appropriate to describe the development of this application because this is out of the scope of this paper. Readers are referred to Ejs' references (Dormido et al., 2005b; Esquembre, 2004; Sanchez et al., 2005).

4.1. Main applet generation

After developing the 3D Ejs' application, user only has to select the option ''Add support for WebGL generation". The main applet prepared for collaboration is created and it is ready to be published in public a web server for sharing it. Fig. 5 shows the appearance of the main applet proposed together with the Ejs' options. The JavaScript control ''WebGL" of the applet generated starts the collaborative system and generates the WebGL simulations in the public web server as it is described in the next subsection.

4.2. Dynamic generation of the WebGL simulation

In order to provide a URL address for the components of a virtual class, a dynamic method to generate an HTML page in the public web server was developed. The software layer of this method was explained in Section 3. This subsection describes how to use it from a user point of view.

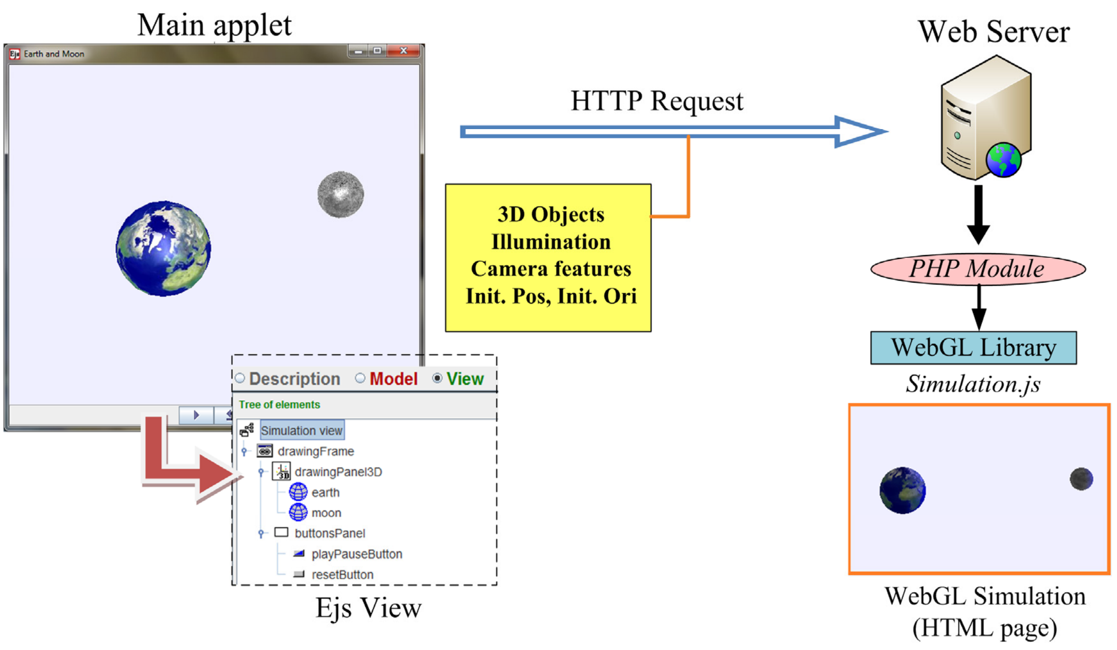
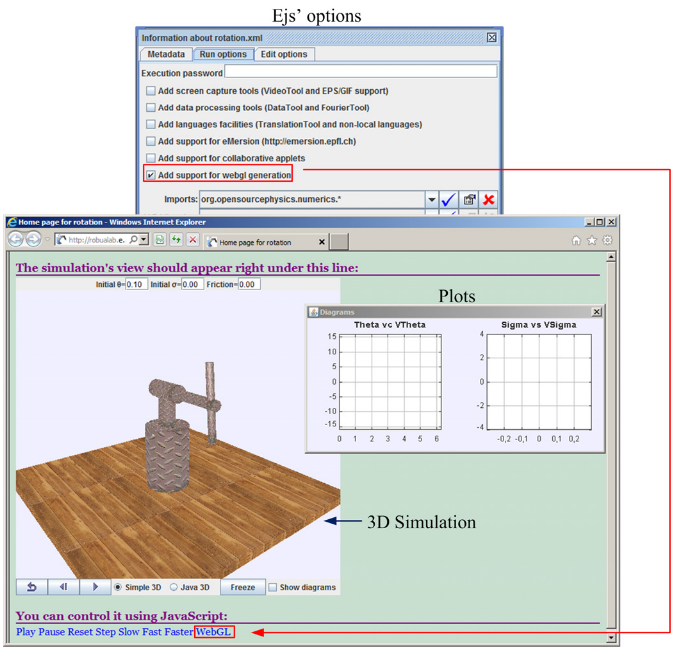


Fig. 4. Generation of WebGL simulation from Ejs' applications

The WebGL simulation's URL is generated directly from the main applet. The JavaScript control ''WebGL'' executes a dialog called *WebGL Generation* where some fields appear (see Fig. 6): *Server URL* and *Name*, fields filled in EJS options. The first one is the URL address of the public web server where the main applet is installed, and the second one is the name for the WebGL simulation generated. After filling these fields, the communication system of the main applet sends this data via HTTP protocol to the PHP module located in the public web server and generates an HTML file with the WebGL simulation embedded. The field *WebGL Generated* will show the URL to be used by students or users to access to the collaborative simulations (see Fig. 6). The user of the main applet only has to send the URL to the other class members of the virtual class by means of a simple auxiliary tool such as e-mail, forum or chat.

If the previous process is successful, the control ''Start Web-Socket Server" will be enabled in the dialog. After activating this control, the main applet is listening to WebGL simulation's requests on the port opened. After that, users can connect to a collaborative virtual environment with only one URL address. They have to connect to the URL provided and to click on the control Connect to Ejs-Sim Server" (Fig. 6).

The communication engine of the WebGL simulation sends a Web-Socket request to the remote main applet. Then, the WebGL simulation gets on-line with the main applet and joins in the collaborative virtual session. In this way, users can create a collaborative environment from anywhere at any time following the previously described steps. They only need an Internet connection and a Java-enabled web browser to create a virtual class based on virtual laboratories developed with Ejs.



Execution password

☐ Add screen capture tools (VideoTool and EPS/GIF support)
☐ Add data processing tools (DataTool and FourierTool)
☐ Add languages facilities (TranslationTool and non-local languages)
☐ Add support for eMersion (http://emersion.epfl.ch)
☐ Add support for collaborative applets
☑ Add support for webgl generation

Imports: org.opensourcephysics.numerics.*

The simulation's view should appear right under this line:

Initial θ= 0.10 Initial σ= 0.00 Friction= 0.00

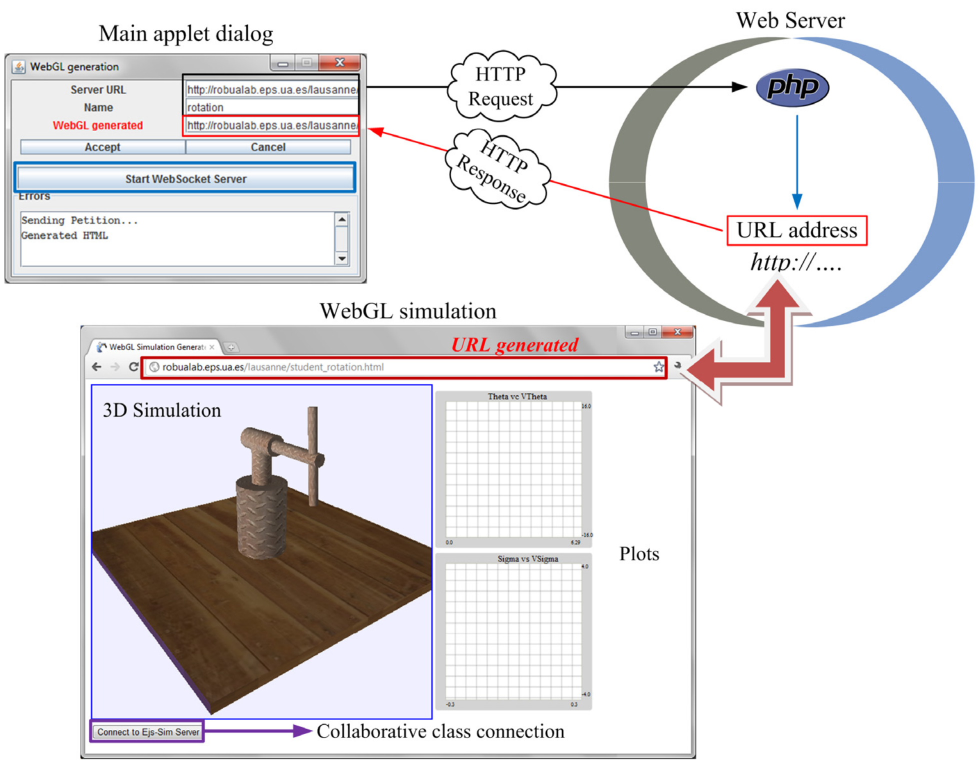
You can control it using JavaScript:

Play Pause Reset Step Slow Fast Faster WebGL

Fig. 5. Main applet generation

4.3. Collaborative virtual class

From the moment of the connection of one WebGL simulation in the virtual class, its user interface is disabled. WebGL users cannot experiment with the shared virtual laboratory and the main applet manages in real-time the simulation evolution. Fig. 7 shows the virtual session of the virtual laboratory proposed where the main applet synchronizes all the WebGL simulations connected to the virtual session. As it can be seen in the Fig. 7, the views or interfaces are in the same state and the main variables are synchronized.

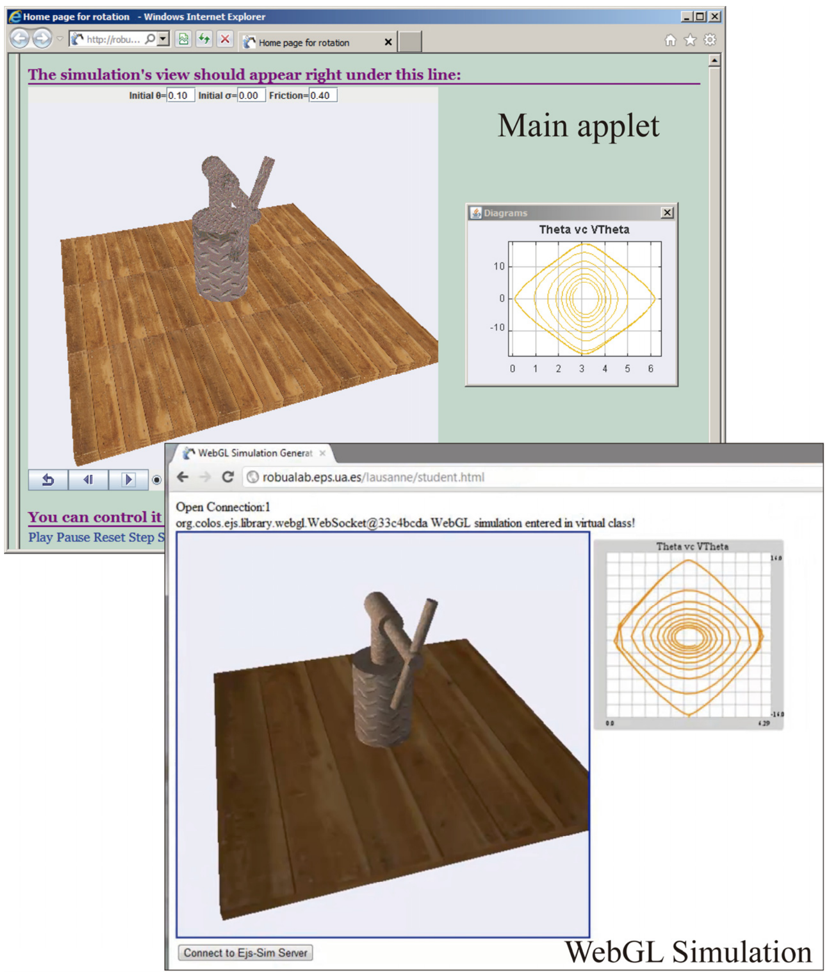


Server URL http://robualab.eps.ua.es/lausanne
Name rotation
WebGL generated http://robualab.eps.ua.es/lausanne

```
Sending Petition...
Generated HTML
```

robualab.eps.ua.es/lausanne/student_rotation.html

Fig. 6. Dynamic generation of the WebGL simulation



Initial θ= 0.10 Initial σ= 0.00 Friction= 0.40

robualab.eps.ua.es/lausanne/student.html

```
Open Connection:1
org.colos.ejs.library.webgl.WebSocket@33c4bcda WebGL simulation entered in virtual class!
```

Fig. 7. Synchronized virtual laboratories (main applet-WebGL simulation) in a virtual collaborative class

5. CONCLUSIONS AND FUTURE WORKS

In this paper, a new web-learning system which combines two outstanding educational resources, the 3D virtual laboratories and the synchronous collaborative learning practice, has been presented. With this approach, a new method to share knowledge in a synchronous way based on experiences over virtual laboratories has been achieved.

This e-learning system represents a portable collaboration framework for simulations developed in Ejs. The communication architecture has been integrated in a new EJS version and can be applied in a transparent and easy way to all the 3D applications developed in this platform. Thanks to the collaborative features that the new Ejs version includes into the applets, users can create a collaborative environment from anywhere at any time by means of the WebGL technology.

As future works, this approach presented in the paper can be extended to develop a complete library for WebGL view applications. This framework can use to separate completely the model and view in the Ejs' environment. This way, several WebGL applications can be refreshed with data from a model which is executing in another computer machine.

ACKNOWLEDGEMENTS

This work has been supported by the following projects and grants: the project DPI2008-02647 of the Spanish Ministry of Education and Science, the GITE grant of the Technology and Educational Innovation Vice-president Office of the University of Alicante, the project 08667/PI/08 of the Fundación Séneca of the Region of Murcia, the projects DPI 2007-61068 and MTM2008-03679/MTM of the Spanish Ministry of Education and Science and the Spanish CICYT under grant DPI2007-61068.

REFERENCES

Alavi, M. (1994). Computer-Mediated Collaborative Learning: An Empirical Evaluation. *MIS Quarterly*, 18, 159-174.

Bafoustou, G., and Mentzas, G. (2002). Review and functional classification of collaborative systems. *International Journal on Information Management*, 22, 281-305.

Dormido, S. (2004). Control learning: Present and Future. *Annual Reviews in Control*, 28, 115-136.

Dormido, S., Dormido R., Sanchez, J. and Duro, N. (2005a). The role of interactivity in control learning. *International Journal of Engineering Education*, 21, 1122-1133.

Dormido, S., Farias, G., Sanchez, J. and Esquembre, F. (2005b). Adding interactivity to existing Simulink models using Easy Java Simulations. In *Proceedings of 44th IEEE European Conference on Decision and Control*, 4163–4168. Sevilla (Spain).

Esquembre, F. (2004). Easy Java Simulations: A software tool to create scientific simulations in Java. *Computer Physics Communications*, 156(2), 199–204.

Gillet, D., Nguyen A.V. and Rekik, Y. (2005). Collaborative web-based experimentation in flexible engineering education. *IEEE Transactions on Education*, 48, 696-704.

Gillet, D. (2010). Tackling Engineering Education Research Challenges: Web 2.0 Social Software for Personal Learning. *International Journal of Engineering Education*, 26(5), 1134-1143.

Jara, C.A., Candelas, F.A., Torres, F., Dormido, S., Esquembre, F. and Reinoso, O. (2009). Real-time collaboration of virtual laboratories through the Internet. *Computers & Education*, 52 (1), 126-140.

Jara, C. A., Esquembre, F., Christian, W., Candelas, F. A., Torres, F. and Dormido, S. (2011). A new 3D visualization framework base on physics principles. *Computer Physics Communications*. In press. DOI: 10.1016/j.cpc.2011.08.007.

Kamel, M., Taylor, A. and Breton, A. (2005). A synchronous communication experiment within an on-line distance learning program: A case study. *Telemedicine Journal and e-Health*, 11, 583-593.

Koschmann, K. (1994). Toward a theory of computer support for collaborative learning. *Journal of the Learning Sciences*, 3(3), 219-225.

Salzmann, C. and Gillet, D. (2011). Remote Labs and Social Media: Agile Aggregation and Exploitation in Higher Engineering Education. In *Proceedings of 2nd IEEE Int. Conf. on Engineering Education*, 110-115, Amman (Jordan).

Sanchez, J., Esquembre, F., Martin, C., Dormido, S., Dormido-Canto, S., Canto, R., et al. (2005). Easy Java Simulations: an open-source tool to develop interactive virtual laboratories using MATLAB/Simulink. *International Journal of Engineering Education*, 21(5), 789–813.

The Web-Sockets API (2011). Available on-line: http://dev.w3.org/html5/websockets/.

Vargas, H.; Sánchez, J.; Jara, C.A.; Torres, F.; Dormido, S. Candelas, F.A. (2011). A Network of Automatic Control Web-Based Laboratories. *IEEE Transactions on Learning Technologies*, 4(3), 197-208.

WebGL - OpenGL ES 2.0 for the Web (2011). Available on-line: http://www.khronos.org/.