REALISTIC MULTIMEDIA SIMULATIONS FOR INFORMATICS STUDENTS

Ioannis Pachoulakis

Department of Applied Informatics and Multimedia, Technological Educational Institute of Crete, Heraklion-Crete, Greece
ip@epp.teicrete.gr

ABSTRACT

Realistic multimedia simulations are effective in helping students overcome their fear of physics and gain fundamental knowledge of physical processes. An elective course has been designed in the Applied Informatics and Multimedia Department at TEI of Crete to help informatics students overcome their physics shyness by hands-on experience on scientific multimedia simulations. The approach is justified in terms of the rich employment opportunities in the game and multimedia industries where a sound basis in physics, mathematics and numerical analysis is a necessity. Student feedback shows that they embrace the adopted approach, which uses open source tools to minimize programming so as to allow both instructor and students to focus on the science and complete a greater number of simulations.

KEYWORDS

Multimedia Simulations, physics, Easy Java Simulations

1. INTRODUCTION

Computational methods and tools are typically not integrated in university science curricula. As a result, new approaches have surfaced [1] to enhance computational modeling activities in science which can fruitfully be adopted in tertiary education. Scientific multimedia simulations offer important exploratory and interactive computational modeling opportunities which effectively help students gain deeper knowledge of the studied processes. They also employ unique animation characteristics to support the real-time visualization of dynamic physical phenomena and processes, as well as the possibility to isolate and study specific interactions. An elective course in our department employs simulations to augment introductory mechanics education. The approach is justified in terms of the significant employment opportunities offered in the game and educational multimedia industries, where a sound understanding of physics, mathematics and numerical analysis is required.

The adopted platform is Easy Java Simulations (EJS), a freely available [2], open-source and well-documented [3], [4] software tool for science students and instructors which eases the rapid prototyping of scientific simulations with high-level graphical capabilities and an increased degree of interactivity. EJS provides well-structured panels to (a) prescribe the time evolution of a model, (b) declare and initialize pertinent variables and (c) assemble the user interface (UI) of the simulation in a drag-and-drop fashion from a rich selection of 2D and 3D components to visualize model state during evolution. In addition, objects from the UI of the simulation are easily and intuitively linked to model dynamic variables in a bidirectional manner which allows both model evolution being directly reflected on the interface in real time and also user intervention, e.g., to change model state by dragging UI components, changing variables or pressing buttons. Simulations are saved as platform-specific XML files. Compiling simulations through the EJS

DOI : 10.5121/ijma.2012.4401

interface creates full-fledged Java applications which can be subsequently packaged and distributed to run in standalone mode, or published on a web server as applets.

The fully worked out simulation presented in Section 2 shows that, for informatics students, EJS seems to strike a perfect balance between programming and playing with physics, as it requires minimal programming and allows more hands-on experience with simulations. Indeed, student feedback data (Section 3) seem to validate EJS as a satisfactory authoring tool for scientific simulations because of its non steep learning curve for informatics students, a fact that allows focusing on the science instead of the tools. As a result, the course may effectively alleviate physics anxiety for many students in the class.

EJS has been usefully paired with other well-known modeling packages such as Matlab and Simulink to create web-based virtual laboratories in a range of fields. For example, EJS has been coupled to MATLAB/Simulink to develop control virtual labs [5], [6] and to study the effect of network delays in control systems [7]. The interactive simulation tool of [8] used to teach 3D Kinematics in robotic arms allows the definition of Denavit–Hartenberg parameters and geometry in serial robotic arms with up to 5 degrees of freedom and provides an innovative graphic interface to study DH convention, observe the movement of serial robotic manipulators and visually map both forward and joint workspaces. RobUALab is a virtual remote robotic laboratory [9] which lets students in automatics and robotics courses complete their practical exercises remotely after having first experimented with the actual plant. In addition, the interactive Java software platform of [10] enables the creation of advanced virtual laboratories with interactive 3D GUIs in the field of robotics. The interactive motion training package of [11] and [12] teaches the basics of motion control of DC and stepper motors as well as robotic arms using theory, movies, dynamic simulators, games and a remote lab. The two-layer approach of [13] employs an experimentation layer which combines EJS with LabVIEW and an e-learning layer which uses eMersion environment to support a flexible educational scheme.

Sample web-based laboratories following this approach are also discussed in [14] and [15]. The real-time collaborative virtual labs of [16], [17] provide synchronized communication of EJS-generated Java applets to enable sharing practical experiences among educators and students in an e-learning synchronous environment. To encourage the study and facilitate the process of autonomous learning in the field of electrical machine learning, [18] develop EJS-based virtual laboratories which facilitating the testing and understanding of these machines. They also discuss the virtues of hands-on and simulated labs approaches and the usefulness of each in their field. A comparative study of the effectiveness of hyper-realistic virtual simulations, traditional schematic simulations and traditional laboratory appears in [19]. The Learning Management System of [20] employs virtual laboratories and an automatic booking system to allow students from different universities to run experiments in the field of control engineering. In the field of Biomedical Engineering, EJS has been coupled to MATLAB/Simulink to understand the operation of the respiratory system under normal conditions and pathological situations, and to predict respiratory variables at different levels of ventilator stimuli and conditions [21].

2. SCIENTIFIC SIMULATIONS WITH EJS: THE MASS AND TWO SPRINGS MODEL

The present section shows how a representative simulation is worked out in class, which comprises a theoretical and a laboratory session. In the theoretical part of the class, the instructor first describes the problem, draws a schematic setup and works out the physics in detail (this is called Stage A). Because our students have very diverse physics and mathematical backgrounds, it is very important to work out the physics in detail, especially in the first few lectures. Furthermore, it is made clear that working out the physics in Stage A is not required for exams.

This decision produces a very positive effect in alleviating physics anxiety for most of the class and, interestingly, makes students more attentive.

Following the complete physics workout, Stage B commences to parse the results of Stage A into appropriate panels of EJS. The platform is quite rich in the range of simulations it can handle and an appropriate sequence of simulations has been selected to familiarize students with the different user interface panels and their intended usage. The simulation described in the present section is presented early in the semester and was constructed using the latest version of EJS (presently v4.3.7) obtained from [2]. In the laboratory part of the class, students are handed the results of Stage A and are asked to work out Stage B for themselves in groups of two.

The schematic of the problem appears in Fig.1. A mass m is attached on two springs of ideal springs S_1 and S_2 with equilibrium lengths (i.e., when not stretched or compressed) L_1, L_2 and spring constants k_1 and k_2, respectively. The other ends of the springs are attached to unmovable walls at positions (0,0) for S_1 and at (X,Y) for S_2.

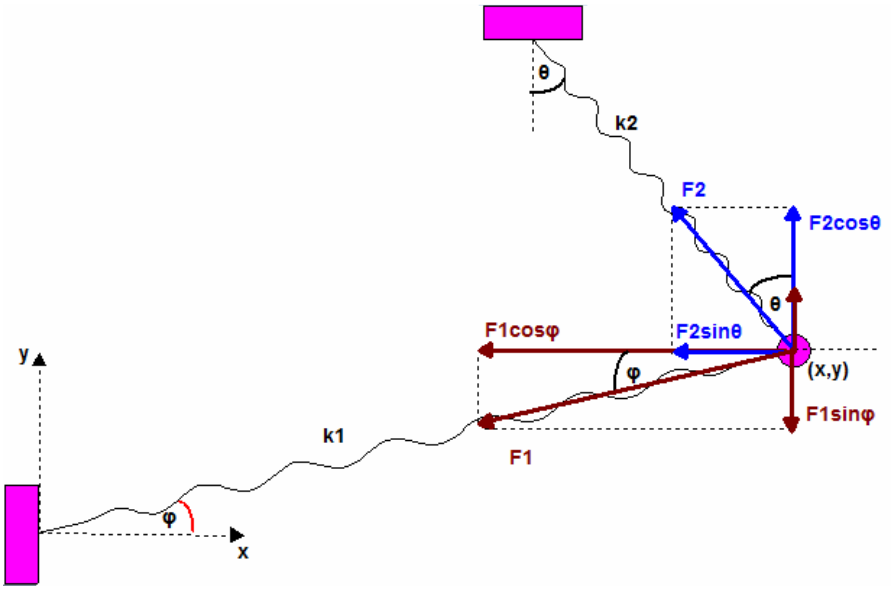

Figure 1. The setup for the mass and two springs model

Stage A: Working out the physics

The mathematical representation of the problem is as follows. At some time t, assume that springs S_1 and S_2 are extended to lengths l_1 and l_2, respectively, as shown in Fig.1. The restoring forces F_1 and F_2 that act on the mass are directed backwards, along the length of the springs and in the direction toward the fastened origin of each spring. In addition, the magnitudes of the forces are given by Hooke's Law:

$$F_1(l_1) = -k_1(l_1 - L_1) \quad (1a)$$
$$F_2(l_2) = -k_2(l_2 - L_2) \quad (1b)$$

From the geometry of Fig.1, we obtain:

$$l_1 = \sqrt{x^2 + y^2} \quad (2a)$$

$$l_2 = \sqrt{(x-X)^2 + (y-Y)^2} \quad (2b)$$

and

$$\cos\varphi = x/l_1 \quad (3a)$$

$$\sin\varphi = y/l_1 \quad (3b)$$

$$\cos\theta = (Y-y)/l_2 \quad (3c)$$

$$\sin\theta = (x-X)/l_2 \quad (3d)$$

Applying Newton's third law on the forces exerted on the mass in the x- and y- directions yields:

$$m\frac{dv_x}{dt} = F_x = F_1\cos\varphi + F_2\sin\theta \quad (4a)$$

$$m\frac{dv_y}{dt} = F_y = F_1\sin\varphi - F_2\cos\theta \quad (4b)$$

Substitution of (2a,b) into (1a,b) and (3a-d), plugging the results into (4a,b) and finally dividing both sides of (4a,b) by m yields the following system of first degree ordinary differential equations (ODEs), the first two of which are simply the definition of speed along the x and y axes:

$$\frac{dx}{dt} = v_x \quad (5a)$$

$$\frac{dy}{dt} = v_y \quad (5b)$$

$$\frac{dv_x}{dt} = -\frac{k_1}{m}\left(\sqrt{x^2+y^2} - L_1\right)\frac{x}{\sqrt{x^2+y^2}} - \frac{k_2}{m}\left(\sqrt{(x-X)^2+(y-Y)^2} - L_2\right)\frac{x-X}{\sqrt{(x-X)^2+(y-Y)^2}} \quad (5c)$$

$$\frac{dv_y}{dt} = -\frac{k_1}{m}\left(\sqrt{x^2+y^2} - L_1\right)\frac{y}{\sqrt{x^2+y^2}} - \frac{k_2}{m}\left(\sqrt{(x-X)^2+(y-Y)^2} - L_2\right)\frac{y-Y}{\sqrt{(x-X)^2+(y-Y)^2}} \quad (5d)$$

The system of ODEs (5) expresses the time evolution of the system and can be solved in real time. To complete the physics description, we note that it is very instructive to watch how, as the system evolves, part of the kinetic energy (of the ball) is transformed into dynamic energy (of the springs) and vice versa while at the same time the total energy E of the system remains constant (as should be expected in the absence of external forces, friction, etc). Accordingly, if *T* denotes the kinetic energy of the ball, $V_{1,2}$ the potential energies of the individual springs, *V* the potential energy of the system and *E* the total energy of the system, then:

$$T = \frac{1}{2}m(v_x^2 + v_y^2) \quad (6a)$$

$$V_{1,2} = \frac{1}{2}k_{1,2}(l_{1,2} - L_{1,2})^2 \quad (6b)$$

$$V = V_1 + V_2 \quad (6c)$$

$$E = T + V \quad (6d)$$

This concludes Stage A. It must be noted that this step-by-step derivation is carried out on the whiteboard analytically, especially in the early simulations, as most students have forgotten a

significant part of their physics background. For example, in a typical theoretical session is approximately equally divided between stage A and stage B (directly below).

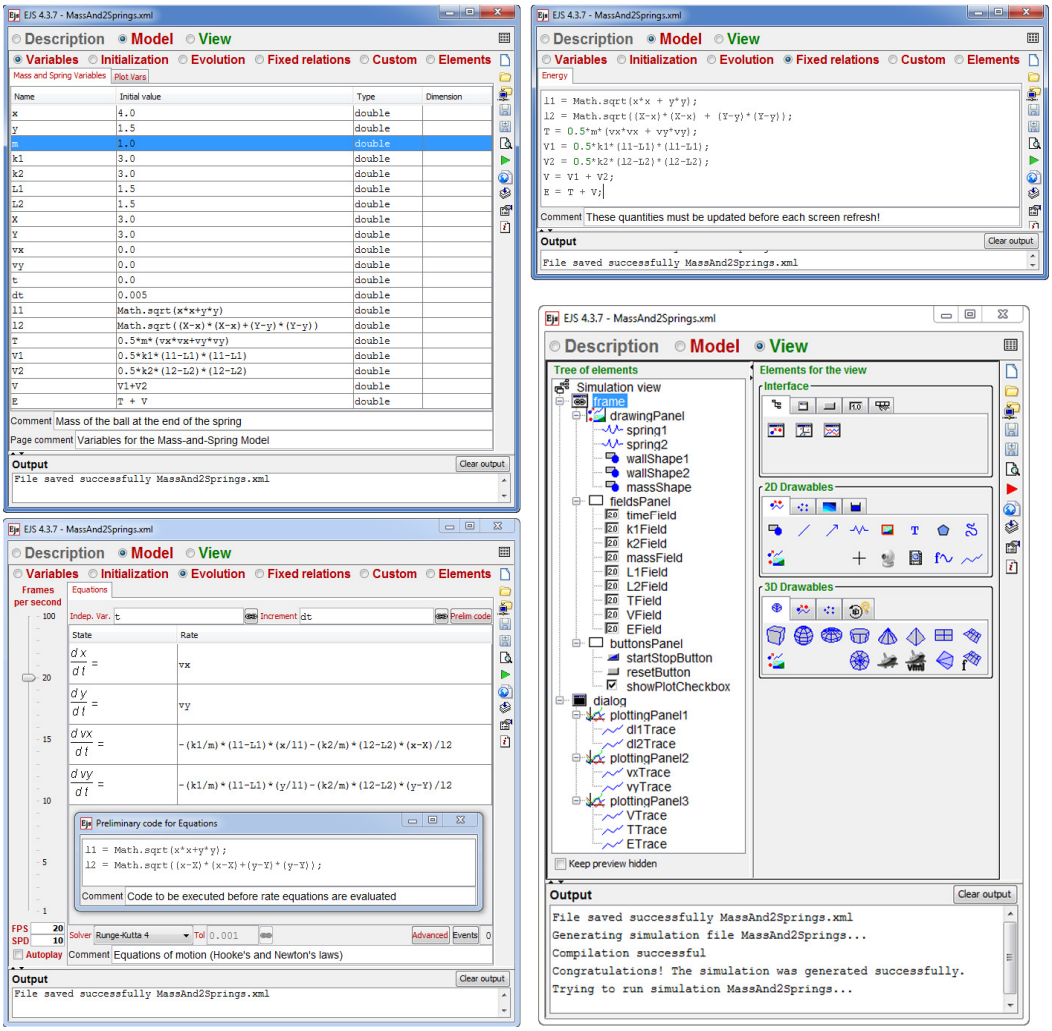

Figure 2. The pertinent panels of the EJS interface for the mass and two springs simulation

Stage B: Constructing the Simulation

After the physics has been derived and explained in Stage A, the simulation is constructed in Stage B. EJS can handle a wide gamut of simulations so only the pertinent panels for the current simulation are discussed below.

The "Variables" panel: Students are reminded that EJS translates the differential equations into Java code internally, so that all necessary variables must be declared and initialized. Direct inspection of (6) and (7) yields the following variables: *x, y, m, k1, k2, L1, L2, X, Y, vx, vy, t, dt, l1, l2, T, V1, V2, V* and *E.* These are declared and initialized in a variables tab named "Mass and Spring Variables", under the menu Model→Variables (top left in Fig. 2).

The "Evolution" panel: The heart of the model is the system of ODEs which governs its evolution. ODEs such as (5a-d) are easily and intuitively parsed into the Model→Evolution panel

of EJS (bottom left in Fig. 2). First of all, the fields "Indep.Var." and "Increment", list the independent variable (t) and increment (dt) for the solver. Then, typing the variables *x, y, vx* and *vy* in the LHS of the ODEs formats the LHS of the equations page nicely as shown. Finally, the RHS of each equation is typed in using standard Java syntax. In order to keep the code typed in the RHS of equations (5c) and (5d) more manageable and at the same time reduce repetition of common calculations in the last two ODEs, we opt to type in the following equivalent set of ODEs.

$$\frac{dx}{dt} = v_x \tag{7a}$$

$$\frac{dy}{dt} = v_y \tag{7b}$$

$$\frac{dv_x}{dt} = -\frac{\mathrm{k_1}}{m}(l_1 - \mathrm{L_1})\frac{\mathrm{x}}{l_1} - \frac{\mathrm{k_2}}{m}(l_2 - \mathrm{L_2})\frac{\mathrm{x-X}}{l_2} \tag{7c}$$

$$\frac{dv_y}{dt} = -\frac{\mathrm{k_1}}{m}(l_1 - \mathrm{L_1})\frac{\mathrm{y}}{l_1} - \frac{\mathrm{k_2}}{m}(l_2 - \mathrm{L_2})\frac{\mathrm{y-Y}}{l_2} \tag{7d}$$

The expressions for variables *l1* and *l2* must, however, be computed before each integration step, so they are inserted in the thoughtfully prescribed EJS "prelim code" screen (seen as a floating window in the bottom left of Fig. 2. Clearly, this makes for more tidy code. The documentation on EJS [2] explains a number of settings which govern the accuracy of the integrated model (e.g., Solver, Tolerance, Advanced options, Events) as well as settings that impact the real time visual evolution of the model (e.g., FPS and SPD). These are fully explained to the students and the effect of different settings on the resulting simulation is demonstrated.

The "Fixed Relations" panel: The ODEs in the Evolution panel calculate updated values for the following dynamic variables: *x, y, vx* and *vy*. However, to keep track of other physical quantities which we want to plot (such as the energies) we need to update their values just before each screen refresh. Accordingly, the corresponding equations (2a,b) and (6a-d) are also entered into the "Fixed Relations" panel (top right in Fig. 2). Note that updated values for variables *l1* and *l2* are computed using (2a,b) prior to computing *V1* and *V2*.

The "View" panel: This panel allows the simulation author to construct the user interface of the simulation itself in a tree structure under the root "Simulation view" (bottom right in Fig. 2). A rich selection of 2D and 3D elements allows for an extensive variety of simulations. The simulation for the Mass and two Springs model requires the following elements:

- frame: A top-level window which hosts the following drawables:
    - drawingPanel: A 2D container for drawables such as the walls, springs and mass.
        - spring1, spring2: objects of type Spring (2D spring). The endpoints of each spring are hooked to appropriate model variables. For example, spring1 has origin at (0,0) and endpoint at (x,y), whereas spring2 has origin at (X,Y) and endpoint also at (x,y).
        - wallShape1, wallShape2: objects of type Shape which represent the fixed walls.
        - massShape: an additional object of type Shape which represents the mass. Its coordinates are set to (x,y), to allow updating its position in real time as the model evolves.
    - fieldsPanel: contains textboxes for the following quantities:
        - editable textboxes for the following variables: *k1, k2, m, L1, L2*.
        - non-editable textboxes for for the following variables: *t, T, V , E*.
    - buttonsPanel: hosts the following drawables:

- startStopButton: a two-state Play/Pause button.
- resetButton: a button of type Button for reseting the simulation.
- showPlotCheckbox: checkbox to make the plots in the second window (dialog, see below) visible or invisible.

- dialog: An object of type Dialog which hosts the following three plotting panels, each with its own set of axes:
  - plottingPanel1: contains two plots of type Trace (a sequence of points):
    - dl1Trace: extension of spring S1: $l1 - L1$.
    - dl2Trace: extension of spring S2: $l2 - L2$.
  - plottingPanel2: contains the following plots:
    - vxTrace: the x-component of speed vx.
    - vyTrace: the y-component of speed vy.
  - plottingPanel3: contains three plots for the energies:
    - VTrace: the potential energy V.
    - TTrace: the kinetic energy T.
    - ETrace: the total energy E.

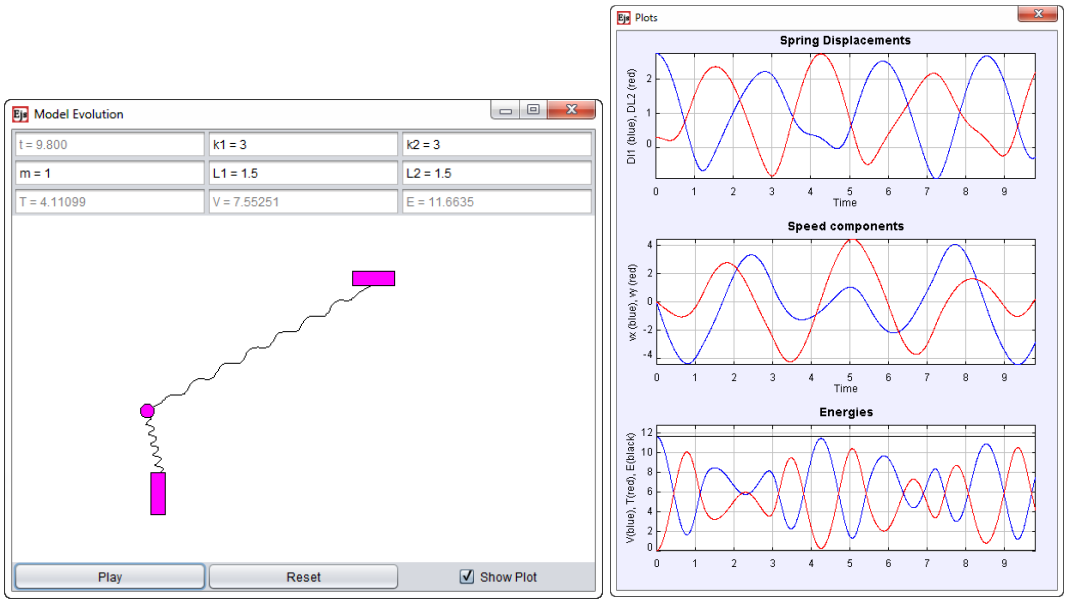

Figure 3. Snapshot of a paused running simulation showing the current state of the model on the left as well as plots of the evolution of pertinent physical parameters on the right.

Finally, Fig. 3 shows snapshots of a paused running simulation. On the left subfigure we can see the current status of the model with the values of pertinent variables at its top. The bottom row of buttons allows for running/pausing and resetting the simulation. The checkbox "Show Plot" controls the visibility of the window on the right, which is a snapshot of the Plots window showing the evolution of various physical quantities: spring displacements $l1 - L1$ and $l2 - L2$ on the top plot, vx and vy on the middle plot and potential energy V, kinetic energy T and total energy E in the bottom plot. Note that the total energy remains flat, as it should in the absence of friction or other external forces.

3. STUDENT FEEDBACK AND CONCLUSIONS

To assess the degree to which the course attains its main goals (expose students to scientific simulations and alleviate their physics-shyness), a questionnaire was distributed to the class approximately half-way into the spring semester of the academic year 2011-12. The questionnaire contained two groups of questions of which the first relates to the ease of use of EJS as a tool for scientific simulations (Table 1) and the second assesses how helpful the specific class structure is for students (Table 2).

Table 1. Student feedback on the ease of use of the platform.

Number of students that answered in each bin 1 through 5							
How easy is each of the following (1=very difficult, 5=very easy)	***1***	***2***	***3***	***4***	***5***	***MEAN***	***STD***
JRE installation	-	-	6	5	20	4.5	2.1
EJS installation	-	2	5	8	17	4.3	2.6
Declare and initialize vars	-	1	10	11	10	3.9	2.5
Enter the time-dependent ODEs	-	6	8	13	5	3.5	2.9
Track constraint variables	-	5	13	10	4	3.4	2.7
Assemble the interface	-	2	12	10	8	3.8	2.6
Couple UI elements to model vars	-	3	12	14	2	3.5	2.3
Running the simulation	-	-	3	13	15	4.4	1.7
Handle settings like FPS, ODE solution method, etc.	-	4	14	8	4	3.4	2.6
Package and distribute simulations	1	8	8	8	2	3.1	3.0

The first two questions in Table 1 address the ease of installation of the (freely downloadable) Java Runtime Environment and of EJS itself. They function as control questions to assess how many students may have trouble even with fundamental tasks such as these. The remaining questions address the ease of use of individual EJS panels, of the pairing of UI drawables to model variables and of the simulation packaging and distribution facilities provided by EJS. The rather low feedback numbers for the latter agree with the author's expectation, since the packaging and distribution capabilities of EJS were demonstrated only once in the first two session of the course. In addition, in an open-ended question asking for suggestions to improve the theoretical and the laboratory parts of the course, most students seemed satisfied, while a few asked for more examples.

Table 2. Student feedback on class architecture.

Number of students that answered in each bin 1 through 5							
How helpful is: (1=not at all, 5=very much)	***1***	***2***	***3***	***4***	***5***	***MEAN***	***STD***
Stage A: Where the teacher works out all the physics and demos the simulation (theory)	2	2	12	11	2	3.3	2.8
Stage B: Where students reconstruct the simulation in pairs (laboratory)	-	2	13	12	4	3.6	2.3
Grading Stage B, but not Stage A, removes physics anxiety	-	3	7	11	10	3.9	2.7
EJS helps me understand Mechanics	2	7	8	5	10	3.4	4.0

The feedback results on the first two questions in Table 2 supports the view that students are agreeable to tools such as EJS which allow them to put most of their effort on creating simulations instead of dealing with programming details, although the author expected a more positive response. Still, feedback relating to the third question seems to justify the approach of not penalizing for physics-shyness, but instead rewarding for active participation into the simulation construction and running. Finally, based on the feedback to the last question, students expect that introducing EJS in their freshman physics course may have a significant effect in their better understanding introductory Mechanics. A similar course of action has been followed by [22] in their course "Dynamic Systems Simulation", adapted for engineering students, with encouraging results.

Based on the feedback results, some corrective measures will have to take place in the second half of the semester so that students have a full exposition to the capabilities of EJS. The platform is a very satisfactory authoring tool for scientific simulations because its non steep learning curve (at least for informatics students) allows them to focus on the science instead of the tools. Finally, the questionnaire will be redistributed at the end of the semester to assess how students' perceptions about physics, simulations and EJS have changed. These results may instructively be cross-correlated to student academic performance in general and in freshman physics in particular.

ACKNOWLEDGEMENTS

Work in this paper has been partially funded by the European Union and the Hellenic General Secretariat for Research and Technology (GSRT) under the "COOPERATION 2009 - 09SYN-72-956" Framework.

REFERENCES

[1] R. G. Neves, J. C. Silva, and V. D. Teodoro, "Improving learning in science and mathematics with exploratory and interactive computational modelling", Trends in Teaching and Learning of Mathematical Modelling, International Perspectives on the Teaching and Learning of Mathematical Modelling, Vol. 1, pp331–339, 2011.

[2] F. Esquembre, "Easy java simulations", Last accessed 5 June 2012 at http://www.um.es/ fem/EjsWiki/Main/Download/

[3] F. Esquembre "Easy java simulations: a software tool to create scientific simulations in java", Computer Physics Communications, Vol. 156, pp199–204, 2004.

[4] W. Christian and F. Esquembre, "Modeling physics with easy java simulations", The Physics Teacher, Vol. 45, pp475–480, 2007.

[5] J. Sanchez, F. Esquembre, C. Martın, S. Dormido-Canto, R. Pastor, and A. Urquıa, "Easy java simulations: an open-source tool to develop interactive virtual laboratories using matlab/simulink," Int. J. Engng Ed, Vol. 21, pp798–813, 2005.

[6] E. Fabregas, G. Farias, S. Dormido-Canto, S. Dormido, and F. Esquembre, "Developing a remote laboratory for engineering education," Computers & Education, Vol. 57, pp1686-1697, 2011.

[7] G. Farias, R. D. Keyser, S. Dormido, and F. Esquembre, "Developing networked control labs: A matlab and easy java simulations approach," IEEE Trans. Ind. Electron., Vol. 57, pp3266–3275, 2010.

[8] M. T. J. Sanguino and A. J. M. Marquez, "Simulation tool for teaching and learning 3d kinematics workspaces of serial robotic arms with up to 5-dof," Computer Applications in Engineering Education, pp1-12, 2010.

[9] 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," Computers & Education, Vol. 57, pp2451-2461, 2011.

[10] C. A. Jara, F. A. Candelas, J. Pomares, and F. Torres, "Java software platform for the development of advanced robotic virtual laboratories," Computer Applications in Engineering Education, Vol. 156, pp1-17, 2011.

[11] C. Buiu, "An integrated internet-based package for teaching motion control: Content and testing results," in Proc. 17th World Congress International Federation of Automatic Control (IFAC'08), Seoul, Korea, pp9797–9801, 2008.

[12] C. Buiu, "Design and evaluation of an integrated online motion control training package," IEEE Trans. Educ., Vol. 52, pp385–393, 2009.

[13] H. Vargas, J. Sanchez, N. Duro, R. Dormido, S. Dormido-Canto, G. Farias, S. Dormido, F. Esquembre, C. Salzmann, and D. Gillet, "A systematic two-layer approach to develop web-based experimentation environments for control engineering education," Intelligent Automation and Soft Computing, Vol. 14, pp505–524, 2008.

[14] S. Dormido, H. Vargas, J. Sanchez, R. Dormido, N. Duro, S. Dormido-Canto, and F. Morilla, "Developing and implementing virtual and remote labs for control education: The uned pilot experience," in Proc. 17th World Congress International Federation of Automatic Control (IFAC'08), Seoul, Korea, pp8159–8164, 2008.

[15] S. Dormido, J. Sanchez-Moreno, H. Vargas, L. de la Torre, and R. Heradio, UNED Labs: a Network of Virtual and Remote Laboratories. Spain: University of Deusto Bilbao, pp273–270, 2011.

[16] C. A. Jara, F. A. Candelas, F. Torres, S. Dormido, F. Esquembre, and O. Reinoso, "Real-time collaboration of virtual laboratories through the internet," Computers & Education, Vol. 52, pp126–140, 2009.

[17] C. A. Jara, F. A. Candelas, F. Torres, S. Dormido, and F. Esquembre, "Synchronous collaboration of virtual and remote laboratories," Computer Applications in Engineering Education, Vol. 20, pp124–136, 2012.

[18] P. Casals-Torrens and R. Bosch-Tous, "Virtual labs for learning electrical machines in marine engineering," in Proc. 3rd International Conference on Maritime and Naval Science and Engineering (MN'10), Constantza, Romania, pp108–112, 2010.

[19] G. Martınez, F. L. Naranjo, A. L. Perez, M. I. Suero, and P. J. Pardo, "Comparative study of the effectiveness of three learning environments: Hyperrealistic virtual simulations, traditional schematic simulations and traditional laboratory," Physics Education Research, Vol. 7, pp1-12, 2011.

[20] H. Vargas, J. Sanchez, C. A. Jara, F. A. Candelas, F. Torres, and S. Dormido, "A network of automatic control web-based laboratories," IEEE Trans. Learning Technologies, Vol. 4, pp197–208, 2011.

[21] A. M. Hernandez, M. A. M. nanas, and R. Costa-Castello, "Learning respiratory system function in bme studies by means of a virtual laboratory: Respilab," IEEE Trans. Educ., Vol. 51, pp24–34, 2008.

[22] Y. Bolea and A. Grau, "A novel pedagogical tool integrating sustainability competence into engineering degrees," in Proc. 41th Annual frontiers in education: celebrating 41 years of monumental innovations from around the world conference program, Rapid City, South Dakota, pp1–6, 2011.

Authors

Ioannis Pachoulakis received a B.Sc. in Physics (1988) at the University of Crete, Greece, and a Ph.D. in Astrophysics (1996) and an M.Sc. in Engineering (1998), both from the University of Pennsylvania in the U.S.A. Since 2001, he serves as an Assistant Professor at the Department of Applied Informatics and Multimedia at TEI of Crete with mainstream interests in realistic multimedia applications, virtual reality and multimedia applications for science.