IL NUOVO CIMENTO Vol. 33 C, N. 3 Maggio-Giugno 2010
DOI 10.1393/ncc/i2010-10631-0

Colloquia: MPTL14

Easy Java Simulations and the ComPADRE OSP collection(*)

W. Christian(1), F. Esquembre(2) and B. Mason(3)

(1) *Physics Department, Davidson College - Davidson, USA*
(2) *Mathematics Department, University of Murcia - Murcia, Spain*
(3) *Homer L. Dodge Department of Physics and Astronomy, University of Oklahoma Norman, USA*

(ricevuto il 30 Novembre 2009; pubblicato online il 27 Luglio 2010)

Summary. — Current technologies allow physics educators the ability to combine traditional instruction with computational modeling. However, the implementation of a computational physics based modeling pedagogy requires a considerable programming effort for teachers and students who want to use this approach. This paper describes a pedagogy that limits the amount of programming when designing, implementing, distributing, and using computer models. It is based on the integration of the Easy Java Simulations modeling tool with the ComPADRE National Science Digital Library.

PACS 01.50.-i – Educational aids.
PACS 01.50.F- – Audio and visual aids.

– Easy Java Simulations (EJS) is free open source software that is designed to create interactive simulations in Java (applications and applets) without the necessity of prior programming knowledge.

– ComPADRE is an NSF-sponsored national digital library that is a collaboration of the American Association of Physics Teachers, the American Physical Society, and the American Institute of Physics.

Index Terms—computational physics, modeling, modeling cycle, simulation, e-learning, digital library, Java programming.

(*) Partial funding for this work was obtained through NSF grant DUE-0442581 (WC), through the Spanish grant DPI2007-61068 (FE) and through NSF grants DUE-0532798 & DUE-0226129 (BM).

1. – Introduction

Over the past dozen years the Open Source Physics (OSP) project has produced some of the most widely used interactive computer-based curricular materials for the teaching of introductory and advanced physics courses. These materials are based on Java applets called Physlets [Christian 2001] and on new OSP programs and authoring tools [Christian 2007, Gould 2007].

Easy Java Simulations (EJS) expands the OSP tools by providing a free open-source program, developed in Java and designed to streamline the creation of dynamic simulations [Esquembre 2004, Easy Java Simulations 2009]. EJS was originally designed for interactive learning under the supervision of educators but is suitable for use by researchers to prototype computer applications and by authors to develop and distribute Java-based curricular materials. While some programming knowledge is assumed, *EJS* encourages users to focus on modeling rather than on programming.

This paper outlines the pedagogical and technical features of the OSP project and how we use OSP-based resources to introduce modeling into the curriculum. We describe our current effort to create and distribute new material using the Easy Java Simulations modeling and authoring tool and how we use the client-server relationship between this tool and the ComPADRE National Science Digital Library [OSP 2009]. The paper is organized as follows. Section **2** introduces the Modeling Cycle that is the basis for our pedagogy. Section **3** describes basic Easy Java Simulations concepts. The main features of the ComPADRE-EJS connection are presented in sect. **4** and sect. **5** presents the pedagogical benefits of this connection. Finally, sect. **6** summarizes the improvements obtained by the use of our approach.

2. – Modeling cycle

Physics education has become an important research topic in the last few decades [Redish 2003]. The increasing interest in exploring new teaching methods in physics has its roots in: 1) the realization that interactive engagement teaching pedagogies improve learning, and 2) the desire to incorporate current technologies and current professional practice into curricula. Computational physics education has much to gain from the synthesis of these new learning pedagogies and tools [Christian 2008].

The modeling approach to teaching is a research-proven pedagogy that predates computers. It attempts to enhance student achievement through a process called the Modeling Cycle. The Modeling Cycle was pioneered by Robert Karplus [Fuller 2001] and the SCIS Project in the 1960s and 70s and later extended by the Modeling Instruction Program led by Jane Jackson and David Hestenes at Arizona State University [Jackson 2008].

The goal of modeling is to teach in a student-centered environment where students do not solve problems in a formula-centered way. The start of the modeling cycle is the development of the model by:

- Qualitative description
- Identification of variables
- Planning an experiment
- Performing the experiment
- Analysis of the experiment

– Presentation of results

– Generalization

After development, the model is employed to study a variety of new physical situations in a variety of ways to test, expand, and enrich the student-created model.

Although the Modeling Cycle can be used without computers, it is well suited for computer modeling if we replace the word "experiment" with "simulation" in the development phase. The analysis of a computer simulation is, in fact, similar to that of a laboratory and often provides the student with a novel perspective on the behavior of a system. Furthermore, the use of computers allows students to study problems that are very difficult and time consuming to study experimentally, to visualize their results, and to communicate their results with others.

The Modeling Cycle approach has been shown to correct weaknesses of traditional instruction by actively engaging students in the design of physical models that describe, explain, and predict phenomena. It is believed that the combination of computer modeling, theory, and experiment can achieve insight and understanding that cannot be achieved with only one approach.

3. – Easy Java simulations

The architecture of *EJS*, shown in fig. 1, is based on the model-view-control (MVC) design pattern, where a simulation is composed of three parts:

1. The model which describes the physics under study in terms of

– Variables that describe the different possible states of the system.

– Relations among these variables (laws that govern the physics), expressed by a computer algorithm.

2. The control which defines actions that a user can perform on the simulation.

3. The view which shows a graphical representation (either realistic or schematic) of the model and its data.

An EJS View is usually both a view and a control because the graphical user interface displays data and responds to mouse and keyboard actions.

The EJS user interface has three main sections: *Description*, *Model* and *View*. The *Description* section permits the model author to include narrative (such as instructions about how to use the simulation, exercises to carry out, theory, results, etc). The inclusion of this descriptive narrative is an important component of the Modeling Cycle and similar pedagogy.

The *Model* section contains all aspects related to the definition of the physical model, including variables and algorithms. Workpanels within the Model section are selected using radio buttons. The *Variables* workpanel defines tables of global variables that can be used throughout the model. The *Initialization* workpanel contains code pages that are executed after the global variables have been created. The *Evolution* workpanel contains panels of explicit Java code or symbolic ODE rate equations that are executed by an animation timer (thread). The *Fixed relations* workpanel contains code pages that are executed after every evolution step or whenever the user interacts with the model

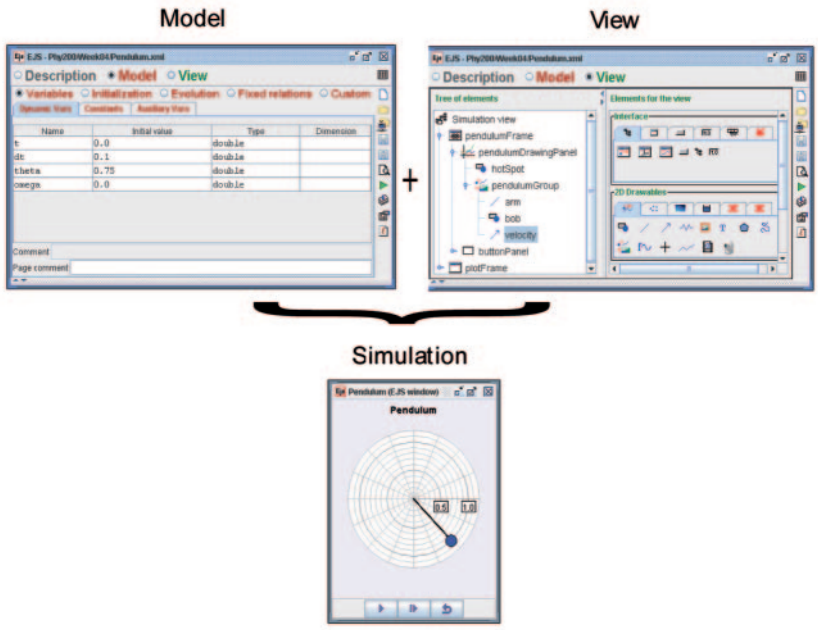

Fig. 1. – Steps to create a simulation in EJS.

using a mouse or keyboard action. The *Custom* workpanel contains code pages with user defined methods (functions).

The Model section's structure may appear restrictive to professional programmers, but this structure is precisely what is needed for the Modeling Cycle. It requires the teachers and students to study and make clear the model's assumptions. For example, a model can implement the Verlet algorithm for the simple physical pendulum within the Evolution workpanel as follows:

```
double a1 = -g*Math.sin(theta)/L;      // initial accel.
theta += omega*dt + 0.5*a1*dt*dt;      // advance position
double a1 = -g*Math.sin(theta)/L;      // final accel.
omega += 0.5*(a1 + a2)*dt;             // advance omega
t += dt;                               // advance time
```

This Evolution workpanel code is the only explicit Java code that appears in this simple teaching model. The global dynamical variables theta, omega, and t and the global constants g, L, and dt are defined in a variables table. (See the EJS tutorial

Fig. 2. – EJS encourages the modeling cycle by organizing the modeling process into well-defined tasks.

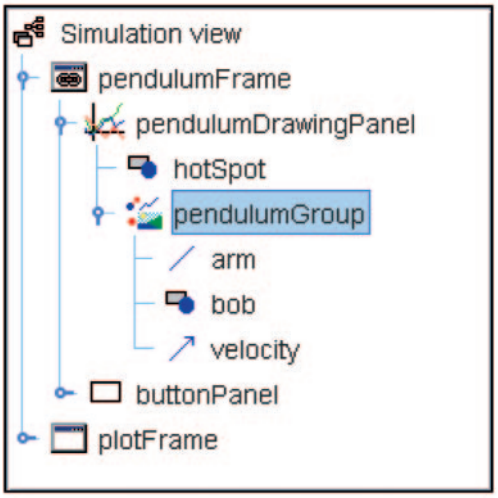

Fig. 3. – The pendulum model's Tree of Elements shows a graphical representation of the view.

available online [Easy Java Simulations 2009] for how to use the symbolic ODE editor rather than an explicit algorithm to solve differential equations and arrays of differential equations.)

The EJS *View* contains the graphical representation of the model's user interface including control elements such as buttons and input fields. The view is built by dragging predefined ready-to-use components from palettes onto a Tree of Elements such as that shown in fig. 3.

EJS *View* components each have an inspector that can be used to set the properties of that model element. Each property can either be a constant value or an expression containing the model variables. Figure 4 displays the inspector for the arrow component that displays the pendulum bob's velocity in the model given above. Its color property is set to MAGENTA, its base is set to $(0, -L)$, and its size is set to omega*L. EJS automatically establishes two-way communication between view properties and model variables so that the view changes as the model evolves and so that the model changes if the user drags an object in the view.

A component's inspector can also contain code that is invoked in response to user actions. Short code fragments can be entered directly into the inspector's action fields. It is good programming practice, however, to define a longer action as a custom method and to invoke this method by entering its name as the action property in the inspector.

The process of creating an advanced simulation is similar to the process shown in fig. 1 for the pendulum model. In the Ising model (see fig. 5) physical parameters such as temperature and field strength and the array of spins are declared in a variables table. The Evolution workpanel contains the 20 lines of Java code needed to flip spins using the Metropolis algorithm. The spins array is an input property for a lattice component that displays the array as color-coded rectangles in the view.

After a model is built, documented, and tested within EJS, the model (including its non-Java resources such as graphics and html description pages) is packaged for distribution into either a jar file or a zip file by clicking on a button within EJS. The resulting jar file, about 1 Mbyte for the pendulum model above, is a stand-alone Java application that

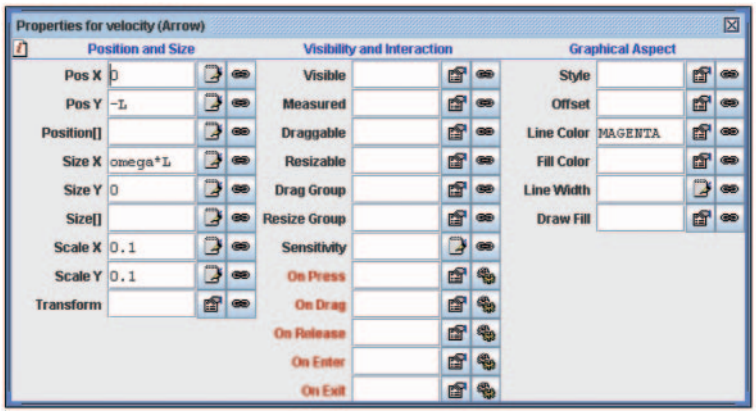

Fig. 4. – An EJS inspector assigns values to an object's properties and invokes methods in response to actions.

does not require EJS and can run on any computer with a Java VM. The model's html documentation appears within a viewing frame and links to PDF documents within the jar open in a native viewer. If a zip package is selected, the resulting 10 Kbyte archive contains the XML source code and other resources but must be unzipped on a computer with EJS to run.

Because an EJS XML source code file is so small, the stand-alone jar file also contains this resource. The immediate availability of the code provides one of the most powerful and exciting aspects of EJS models. Unlike most compiled programs, users can examine, modify, and redistribute the model with minimal effort. Right-clicking within a running simulation displays a pop-up menu with the option to extract the XML file from the jar and to copy it into the local computer's EJS workspace. This packaging trick allows a

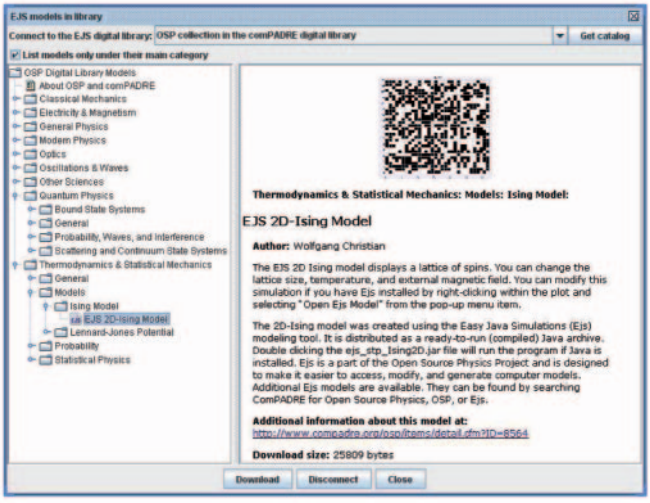

Fig. 5. – EJS models in the OSP Collection in ComPADRE are accessible from within EJS.

teacher to ask students to modify a compiled model and repackage it thereby creating a teacher-student feedback loop that supports the Modeling Cycle.

4. – ComPADRE Digital Library

The ComPADRE National Science Digital Library Pathway [ComPADRE 2009] is a growing network of online collections containing educational resources supporting teachers and students in Physics and Astronomy. Each ComPADRE collection is focused on a particular community (*e.g.*, high school teachers, physics majors, physics education researcher) or topic (*e.g.*, quantum physics, astronomy).

The Open Source Physics (OSP) Collection within ComPADRE contains Java-based computational resources for teaching and the supporting documentation to help teachers and students take advantage of these materials. The goal of the collection is to provide curriculum resources that engage users in physics, computation, and computer modeling. The collection is built on a repository of source code, executable simulations, and curriculum resources. As with any good library, these items are annotated and cataloged so that users can find materials using standard search criteria such as subject, author, and keyword. Other library and web tools are available to the OSP Collection as a part of ComPADRE. These include personal user collections (both private and shared), comments on resources, connections between resources, and the easy incorporation of OSP and EJS pedagogical materials into all other ComPADRE collections.

The collaboration between the OSP and ComPADRE projects has resulted in a new way of sharing EJS models over the web. The EJS modeling environment can act as a client that directly accesses and downloads models in the ComPADRE library. Clicking on the web libraries icon in EJS connects to online repositories and displays a catalog of models in a table of contents as shown in the left pane of fig. 5. Clicking a catalog entry shows a brief description of that model. Double-clicking either the catalog entry or the download button copies the model's XML source code and resources from the library into the local EJS workspace where it can be examined and modified.

Although the OSP Library on ComPADRE is a large central repository of resources, EJS developers have the option of sharing their materials through this same web-client interface. This will give teachers and students access to more resources and also different organizational structures for EJS models. For example, the ComPADRE models are arranged by subject while the Davidson College EJS digital library shows models arranged by course syllabus.

5. – Pedagogical benefits

EJS models in the ComPADRE digital library with associated curricular materials have the following pedagogical benefits:

- *They help students visualize abstract concepts.* The most obvious benefit of simulations in instruction is that they help students visualize systems. In traditional instruction, students learn the concepts of physical science and physics via static pictures and words. Students construct an understanding through internal visualization, which is usually faulty for new topics and will hamper their progress toward understanding the concepts of physics.
- *They are interactive and require student control.* Students often learn to use a "plug-n-chug" approach to problem solving. When faced with end-of-the-chapter

textbook problems, students quickly determine that through a process of elimination, they can find the appropriate equation to use to solve these problems without relying on the physics concepts [Maloney 1994]. In well-designed model-based simulations, physical quantities are not given and must be determined from the simulation. Since determining what information is relevant must be done early in the problem-solving process, students must conceptualize the problem before starting algebraic manipulation.

– *They are more like real-world problems.* Textbook problems are very different from real-world problems. Solving a real-world problem entails distinguishing between relevant and irrelevant information. In a model-based simulation, just as in a laboratory, students must take data that introduces and reinforces the idea that there is uncertainty in measurements and therefore results. This means that two students when faced with the same model-based simulation will end up with slightly different (correct) answers. Using simulations can therefore bridge the gap between theory and the real world [Ronen 2000]

– *They use multiple representations to depict information.* The idea that students learn best when they see the same ideas presented in different ways is not new. Traditional instruction relies on the written and spoken word and other static depictions. Simulations can not only depict motion, but they can also simultaneously depict the information in a different way via graphs and tables that change with time [Van Heuvelen 2001]. In addition, simulations can provide the opportunity to investigate numerous alternate scenarios [Zacharia 2003]

– *They are simple with limited distracting features.* Educational materials are too often developed based on technology with pedagogy as an afterthought. Only graphics, animations, or sounds that contribute to the learning process should be included in a simulation. This allows students to focus in the task without being distracted by unnecessary or overly flashy additions.

– *They can improve assessment of student understanding.* Researchers have shown that simulation-based resources can provide a superior assessment vehicle as compared to traditional paper-based questions. Dancy compared student responses to traditional conceptual exercises with responses to nearly identical simulation-based exercises. She found that in general, the simulation-based version of the exercise was more valid for understanding whether students understood a given concept [Dancy 2002].

6. – Summary

The combination of a computational physics friendly modeling and authoring tool with Internet technologies allows teachers to easily incorporate computer-based modeling into their curriculum by providing an open and extensible solution for the creation and distribution of educational software. EJS is free because it is collaboratively built and released under the GNU GPL software license. ComPADRE has no registration costs because it is part of the National Science Digital Library project and endorsed and supported by the professional societies. EJS produces self-contained teaching modules that are extensible, adaptable, and easily modifiable because the model's documentation and XML description are packaged in the executable jar file. If a model is uploaded into

ComPADRE, its authorship, modifications, and use are documented and intellectually traceable.

The advantage of EJS for computational physics teaching is that it forces students to separate the model into logical parts and to separate the model from the view. Students learn the logic of computer modeling using loops and control structures and study algorithms used in professional practice when building models. Students are also introduced to object-oriented programming concepts by using object properties and methods when they create user interfaces. However, little user-interface coding is required because the user interface is created automatically by EJS.

The Open Source Physics combination of computational physics tools and computer modeling pedagogy with a digital library provides students and teachers with new ways to understand, describe, explain, and predict physical phenomena.

* * *

EJS was created by Francisco Esquembre at the University of Murcia, Spain. The ComPADRE national digital library is funded in part by the NSF and supported by the AAPT, APS, and AIP. Special thanks to L. Barbato the ComPADRE technical director and M. Riggsbee the ComPADRE designer for design of the OSP Collection and helping to implement the EJS client service. The authors gratefully acknowledge that partial funding for this work was obtained through NSF grant DUE-0442581, Davidson College and grant DPI2007-61068 of the Spanish Ministerio de Educación y Ciencia. The ComPADRE Library has been supported by the NSF through grants DUE-0532798 & DUE-0226129.

REFERENCES

Christian W and Belloni M (2001) Physlet Physics, Prentice Hall.
Christian W (2007) Open Source Physics: A User's Guide with Examples, Addison-Wesley.
Gould H, Tobochnik J and Christian W (2007) An Introduction to Computer Simulation Methods : Applications to Physical System 3rd edition, Addison-Wesley.
Esquembre F (2004) Easy Java Simulations: A software tool to create scientific simulations in Java, Comp. Phys. Comm. 156, 199-204.
Easy Java Simulations, `http://www.um.es/fem/Ejs/`, accessed 2009 October.
ComPADRE, `http://www.compadre.org/osp/`, accessed 2009 October.
Redish E F (2003) Teaching Physics with the Physics Suite, Wiley. This book provides an introduction to the cognitive model that underlies much Physics Education Research [PER]. The bibliography references almost all of the seminal papers in PER.
Christian W and Ambrose B editors (2008) Theme Issue on Computation and Computer-Based Instruction, Am. J. of Physics 76 (4&5). This double issue contains 29 articles covering many aspects of computational physics education.
Fuller R editor (2001) A Love of Discovery: Science Education– The Second Career of Robert Karplus, Kluwer Academic.
Jackson J, Dukerich L and Hestenes D (2008) Modeling Instruction: An effective Model for Science Education, Science Education 17 (1), 10-17.
Maloney D P (1994) Research on Problem Solving: Physics in Handbook of Research on Science Teaching and Learning, Gabel, D. (Ed), MacMillan.
Ronen M and Eliahu M (2000) Simulation - A bridge between theory and reality: the case of electric circuits, J. of Computer Assisted Learning, 16, 14-26.

Van Heuvelen A and Zou X (2001) Multiple Representations of Work and Energy Processes, Am. J. Phys. 69, 184-194.
Zacharia Z and Anderson R O (2003), The effects of an interactive computer-based simulation prior to performing a laboratory inquiry-based experiment on students' conceptual understanding of physics, Am. J. Phys. 71 6, 618-629.
Dancy M (2002), Investigating animations for assessment with an animated version of the Force Concept Inventory, Doctoral Dissertation, North Carolina State University.