

Easy Java Simulation: A Modeling Tool for Physics Teaching and Learning

Fu-Kwun Hwang

Dept. of Physics, National Taiwan Normal Univ., Taiwan

Email: hwang@phy.ntnu.edu.tw

Abstract

Easy Java Simulations (Ejs) is designed for teachers and students who want to create (or modify) scientific simulations. With Ejs, they can concentrate their effort in writing and refining the relations in the underlying scientific model, and dedicate the minimum possible amount of time to the programming techniques.

We have found that, by creating a simulation, many teachers get a new perspective of the phenomenon they are trying to explain, which almost always increases their enthusiasm about the use of this technology with their students. An alternative approach, and a very promising one, is to let students modify the model in a simulation or create their own simulations, thus engaging in what educational researchers call constructive modeling.

A computer simulation is a computer program that reproduces a natural phenomenon through the visualization of the evolution of its state. Each state is described by a set of variables that change in time due to the iteration of a given algorithm. Computer simulations in an instructional context involve using the computer to model real-world phenomena in order to help students gain insights into the behavior of complex systems. Computer simulations can help students to understand invisible conceptual worlds of science through animation, which can lead to more abstract understanding of scientific concepts. Quantitative data can be manipulated and visualized to help students form a qualitative mental picture. Such complex experience can help students to identify patterns within simulations, and formulate explanations for phenomena in terms of models and theories. Simulations must not only allow learners to construct and manipulate screen “objects” for exploring underlying concepts, but they must also provide learners with the observation and manipulation tools necessary for exploring and testing hypotheses in the simulated world (Jonassen, 1996). Combined with graphical representations, simulations should allow learner to visualize abstract concepts and to link them to prior knowledge, thereby fostering conceptual learning. Students interacting with an instructional simulation gain a better understanding of a real system, process or phenomenon by exploring concepts, testing hypotheses, and discovering explanations (Lunetta and Hofstein, 1991; Mellar and Bliss, 1993; Raghavan & Glaser, 1995). This interactivity provides opportunities for students to modify their mental models by comparing the outputs of the model with their

expectations (Jackson et al., 1996), and to engage or motivate students to explore and couple actions with effects which will lead to understanding. Creating a simulation by oneself requires an extra effort. The starting point is a full understanding of the phenomenon and physics model being simulated. Normally, beyond an understanding of the physical model, the designer needs the programming and technical expertise to describe the phenomenon in computer language. Easy Java Simulations (Ejs) is a software tool designed for the creation of computer simulations. It is a medium for making, doing, creating and best of all, it can be used as a modeling tool. Construction, simulation, visualization, and analytic description of the physics model are linked during the creation process. With Ejs, the task of creating a simulation is greatly simplified. That is, the majority of the programming work is done by automatic code generation based upon text-based instructions, mathematic formula about the model and menu/mouse selections from the designer creating the graphical user interface for the simulation. Java source code is generated automatically and compiled into class files. In addition, a jar file and the associated html pages are produced for users to view the simulation with a browser. This automatic code generation allows the Ejs user to concentrate on describing the model by defining the parameters related to the model, providing equations for the evolution of those parameters, setting the constraints between variables, and building a graphical representation. An additional advantage of using Ejs is that it causes the designer who is building the simulation to think through the problem in a new way. Ejs was developed for an Open Source Physics Project at the University of Murcia, Spain. Ejs, and the simulations created with it, can be used as independent programs under different operating systems, or be distributed via the internet and run within html pages by most popular web browsers.

What makes Ejs different from most other tools is that Ejs is not designed to make life easier for professional programmers, but instead it was conceived by science teachers, for science teachers and students -- that is, for people who are more interested in the content of the simulation, the simulated phenomenon itself, and much less in the technical aspects needed to build the simulation program. Hence, Ejs provides a conceptual structure and simplified tools that allow concentrate most of designer's time in the description of the model of the phenomenon want to be simulated. The typical audience includes science teachers and researchers who have a basic knowledge of programming, but who cannot afford the big investment of time needed to create a complete graphical simulation. They may be able to describe the phenomena in their respective disciplines in terms of algorithms, but still need an extra effort to create a sophisticated, interactive graphical user interface.

Most computer simulations of scientific phenomena can be described in terms of the model-control-view paradigm. This paradigm states that a simulation is composed of three parts:

1. The **model**, which describes the phenomenon under study in terms of
 - variables, that hold the different possible states of the phenomenon, and
 - relationships among those variables (corresponding to the laws that govern the phenomenon), expressed by computer algorithms.
2. The **control**, which defines certain actions that a user can perform on the simulation.
3. The **view**, which shows a graphical representation of the different states that the phenomenon can have. This representation can be done in a realistic or schematic form.

These three parts are deeply interconnected. The model obviously affects the view, since a change in the state of the model must be made graphically evident to the user. The control affects the model because control actions can (and usually do) modify the value of variables of the model. Finally, the view affects the model and the control, because the graphical interface can contain components that allow the user to modify variables or perform the predefined actions.

To further simplify the construction of a simulation, **Ejs** suppresses the control part, merging it half into the view, half into the model. Actually, modern computer programs are interactive, which means that the user can modify the program's logic by doing some gestures (such as clicking or dragging the mouse, or hitting the keyboard) with the computer peripherals on the program's interface (or view). Thus, the view itself can be used to control the simulation. On the other hand, if we want this interaction to have certain relevance within the program, these gestures on the interface need to trigger actions that affect the model's variables. Therefore, the best place to define these actions is in the model itself.

Creating a simulation in Ejs consists in defining its model and its view (i.e., the graphical user interface/GUI) and establishing the mutual connections needed for

- the correct visualization of the state of the phenomenon being simulated and
- the appropriate interaction of the user with the view (either to modify this state or to perform the actions defined on the model).

This explicit separation in parts reinforces conceptually the central role of the model of a simulation. It is the model which defines what the program simulates and how. There may be different views for a given model. Teachers can create the same simulation with different GUIs for different tasks or different students.

In addition to the Model and View, Ejs has one more component from which a simulation is built– the **Introduction**. For pedagogical or

scientific purposes, it is always helpful to include a description of what a simulation does, including the instructions on how to operate it and other information related to the simulation. This information appears in the **Introduction**, which is used to generate the content of the *html* web page. Therefore, there are three major parts to the interface: **Introduction**, **Model** and **View**.

Figure 1 shows an example of the **Introduction** for a Simple Harmonic Motion (SHM) simulation. Figure 2 shows the SHM simulation that is created with Ejs.

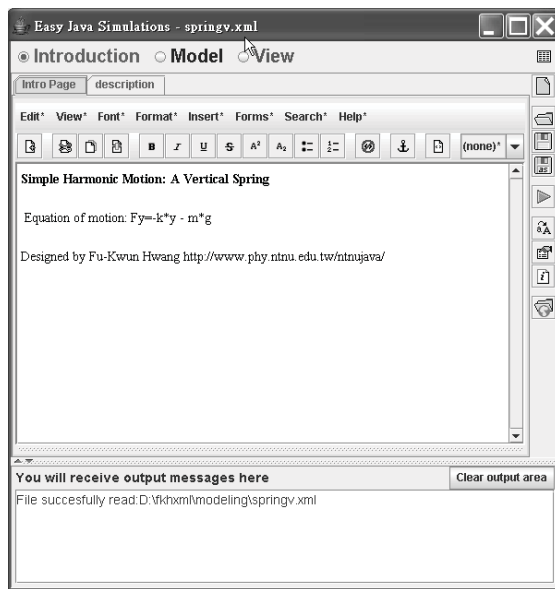


Figure 1: Content in the **Introduction** will be used to generate web pages for the simulation.

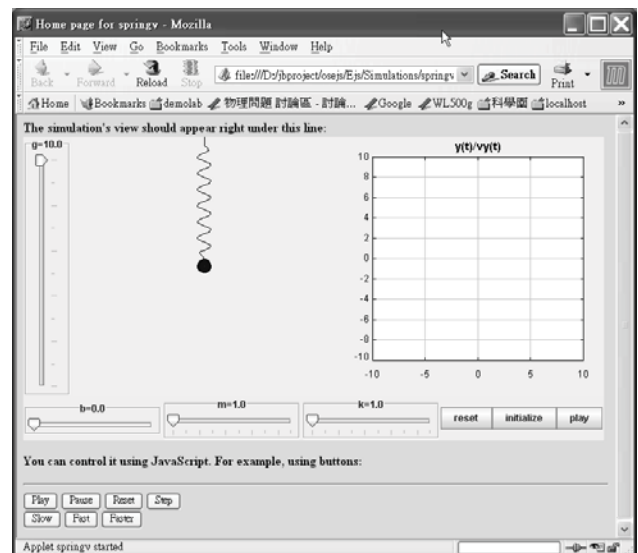


Figure 2: Java simulation embedded in web page created with Ejs.

The **Model** interface follows the **Introduction**, and has five sections:

1. **Variables**: As shown in Figure 3, all the variables for the simulation need to be predefined with specific data types (or dimensions for arrays). The initial value for the variables can be specified here or in the “initialization section”. Users can group the variables into several pages under different names. For example, the SHM simulation has two tabbed pages for the variables. The variables defined in the “coordinate” page are those that specify the boundary for the simulated region on the screen and the timing information in the simulation. The variables defined on the “basic” page are those that correspond to the position and velocity in the X-Y rectangular coordinate system, the mass for the simulated particle, and the spring constant for the SHM, etc.
2. **Initialization** (optional): Different sets of initial conditions can be defined to simulate different cases for the same model (scientific phenomenon). Initial values for the array variables are usually initialized here.

3. **Evolution:** Figure 4 shows that the simulation will be created with 20 frames per second and that the “Runge-kutta 4th order” method is adopted for the numerical integration. The “Independent Variable” is t . When the user enters y in the “State” column and v_y in the “Rate” column, $\frac{dy}{dt}$ is shown instead, in order to specify the evolution condition $\frac{dy}{dt} = v_y$. Notice that the equation for simple harmonic motion is transformed into evolution conditions:
- $$F_y = -k \cdot y - b \cdot v_y - m \cdot g \text{ becomes } \frac{dv_y}{dt} = (-k \cdot y - b \cdot v_y) / m - g,$$
- where m and k are mass and spring constants. Simulations for different models can easily be created by modifying the equations with different forms.
4. **Constraints** (optional): Optional constraints or relations between variables can be defined here.
5. **Custom** (optional): Custom functions can be defined to be used in other sections in the **Model** or as a trigger action for the GUI in the **View**.

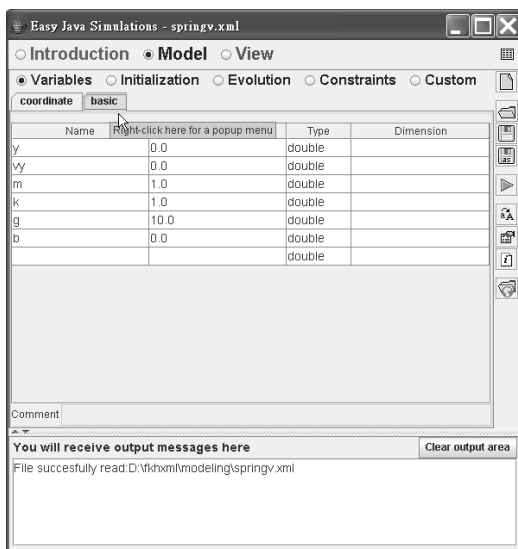


Figure 3: The variables for the SHM simulation.

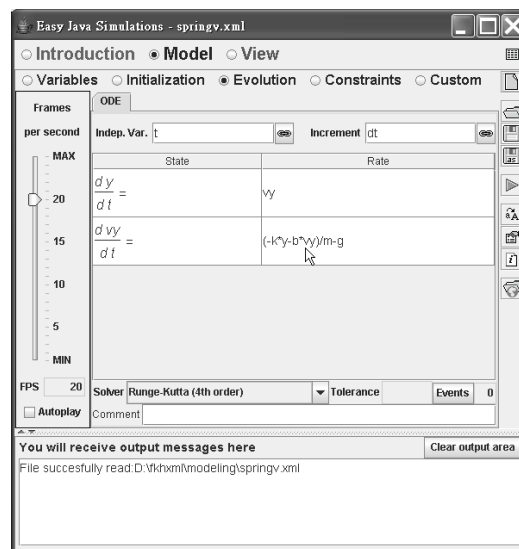


Figure 4: The evolution/constraints for variables.

Physics teachers and students have found it is easy to follow the above steps to specify the mathematics equations in the model for the simulation. These are the same steps they would use to describe SHM in their physics classes. A comparison between the two is shown in Table 1. It is this similarity that makes many science teachers feel, for the first time, they are able to create their own simulations with Ejs.

As shown in Figure 5, Ejs provides an easy-to-use interface to create the GUI for the simulation. Interactive graphic elements can be built with mouse drag and drop actions. The tree-like structure of

elements in the **View** is used to select and edit a particular element. Each element provides some characteristics for itself, called “Properties,”

Steps to describe model for simulation in Ejs	Steps to describe phenomenon in science class
1. Define the boundary for the simulated region.	1. Define the coordinate system.
2. Define variables and give initial values.	2. Specify initial conditions and related parameters
3. Provide evolution and constraints for variables.	3. Provide scientific law or relations.
4. Generate simulation by numerical method.	4. Carry out numerical calculation.

Table 1: Comparisons between steps to describe a model for the simulation in Ejs and steps to describe the same phenomenon in science class.

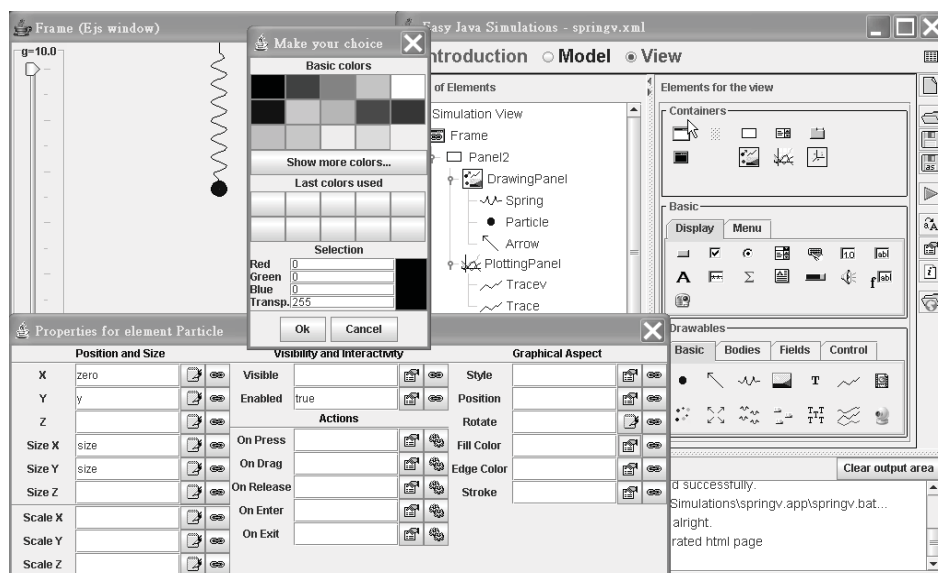


Figure 5: Creating the GUI for the simulation in Ejs.

which can be linked with variables defined in the model. The linking mechanism is a two-way, dynamic process. At any moment, the property of the element will reflect whatever value the linked variable holds, and vice versa. If the property changes as a result of an interaction such as typing in a new value or moving a scrollbar, the variable will receive a new value. For example, the “X, Y, Size X and Size Y” properties for the arrow were set to link with variables x , y , v_x and v_y in the model. Whenever x , y , v_x or v_y are changed with time in the model during the simulation, an arrow in the user interface reflects the change. Also, some

elements have properties of a special type, which are called “Actions,” to specify what to do when the user interacts with the element (e.g. Reset, Pause, Play). Once the **Model** and **View** have been completed, Ejs can generate the Java source code, compile the source to create class files, compress all the class files into a *jar* file and make *html* pages with a single mouse click. The generated simulations will automatically pop up on screen for inspection. If any error occurs when trying to compile the source code, extended information is provided to guide the user to check for the specific pages in the **Model**.

Conclusion and implications:

Simulations can provide insight into the inner workings of a process – not just what happens, but also how and why. We all agree that the computer simulations are not a substitute for observation of and experimentation with real phenomena. Nevertheless, computer modeling with online simulations can add a valuable new dimension to scientific inquiry and understanding. Well-designed computer applications allow learners to use visual and kinesthetic resources to explore phenomena and to test theories so that they may eventually construct a web of connections between new information and information they already know. With Ejas, the task of creating a simulation is greatly simplified. Ejs users concentrate on describing the model by defining the parameters related to the model, providing equations for the evolution of those parameters, setting the constraints between variables, and building a graphical representation --- doing scientific work instead of playing with software. So teachers/students could concentrate most of their effort to teaching/learning instead of technology! Teachers without programming experience have already created simulations for use in their curriculum after an introductory Ejs workshop. We have found that, by creating a simulation, many teachers get a new perspective of the phenomenon they are trying to explain, which almost always increases their enthusiasm about the use of this technology with their students. An alternative approach, and a very promising one, is to let students modify the model in a simulation or create their own simulations, thus engaging in what educational researchers call *constructive modeling*.

Acknowledgement

This work is supported by National Science Council. (NSC94-2511-S-003-007).

References

- Jackson, S. L., Stratford, S. J., Krajcik, J., and Soloway, E., 1996, A learner-centered tool for students building models, *Communications of the ACM*, 39(4), 48-49.
- Jonassen, D. H. (1996). *Computers in the classroom: Mindtools for critical thinking*. Englewood Cliffs, NJ: Prentice-Hall, Inc.
- Lunetta, V., & Hofstein, A. (1991). Simulation and laboratory practical activity. In B. Woodnough (Ed.), *Practical science*, 125-150. Miltonkeynes, PA: Open University Press

Mellar, H. G., & Bliss, J. (1993). Expression the student's concepts versus exploring the teacher's: Issues in the design of mociroworlds for teaching. *Journal of Educational Computing Research*, 9(1), 89-112.

Raghavan, K., & Glaser, R. (1995). Model-Based analysis and reasoning in science: The MARS curriculum. *Science Education*, 79(1), 37-61.

Easy Java Simulations : <http://fem.um.es/Ejs/> and
<http://www.phy.ntnu.edu.tw/ntnujava/>

Open Source Physics Project : <http://www.opensourcephysics.org/>