

## CyberLab: an Online Virtual Laboratory Toolkit for Non-programmers

Kang Zhao

College of Information Sciences and Technology  
The Pennsylvania State University  
University Park, PA 16802, USA  
kangzhao@psu.edu

Matthew P. Evett

Department of Computer Science  
Eastern Michigan University  
Ypsilanti, MI 48197, USA  
mevett@emich.edu

### Abstract

*Online virtual laboratory is important for online science education. CyberLab is a toolkit for online virtual experiments. It is able to handle the creation, exportation, and execution of virtual experiments. We believe CyberLab is the first toolkit that allows instructors, who do not know computer programming, to create their own online virtual experiments. This paper briefly introduces the underlying model and the architecture of CyberLab.*

### 1. Introduction

Online courses have been widely used in science education. However, some theories and equations in science education are very abstract and difficult for students to understand. Instructors of science courses have long recognized the need for laboratory experience, through which students can deepen their understanding of conceptual materials [4]. Therefore, it is necessary and helpful to give students laboratory or experiment experience, especially when they take science courses online.

Although different definitions exist, a virtual laboratory is often described as “a computer simulation which enables essential functions of laboratory experiments to be carried out on a computer” according to [2].

There have been various online virtual laboratories in different science disciplines, such as Virtual Biology Experiments [6], Virtual Chemistry Laboratory [5], and Virtual Reality Physics Simulation [3]. They rely mainly on modeling of knowledge in specific science subjects and programming techniques. Though some of them are very powerful and can be referenced by instructors when teaching online science courses, they suffer from one major disadvantage: instructors cannot modify the experiments to fit their special needs, unless they are the original developer. Even if given source codes to replicate those systems, instructors are still required to be proficient programmers or competent

users of complicated software packages in order to create, customize, and deploy their own online virtual experiments.

Physlets [1], also known as Physics Applets, is a collection of Java applets. According to our research, it is the only existing tool that aims at enabling instructors to create their own online virtual experiments. However, users are still required to know programming with Java Script, which limits its usage among instructors of online science courses.

We thus proposed CyberLab [7], with the purpose of enabling instructors with no programming skills to create their own virtual online experiments. CyberLab is a toolkit for online virtual laboratory. With this toolkit, instructors can easily create virtual experiments and export them into intermediate experiment descriptor files without programming. Students can then download those files from online course websites, reconstruct the experiments designed by instructors and execute them.

### 2. Models of CyberLab virtual experiments

In order to design virtual experiments, it is necessary to represent them with models, so that we can specify how the experiment behaves at certain points. The underlying models for CyberLab virtual experiments includes a collection of Finite State Automata (FSA), independent and dependent variables, and a set of expressions that define the relationships among them.

FSA is the core of the CyberLab experiment model. FSA is chosen mainly because it is an effective and intuitive approach to represent a great number of basic scientific experiments, although we are also aware that it is far from being able to represent all experiments in science education.

In CyberLab, each experiment is represented as a Deterministic FSA. FSA is a model of behavior, which is composed of states, transitions (also known as edges), and actions. A state stores information about the past; i.e., it reflects the input changes from the system's start to the present moment. A transition indicates a state change and is described by a condition that would need to be fulfilled

to enable the transition. An action is a description of an activity that is to be performed at a given moment. For Deterministic FSA, there is a deterministic next state given a pair of current state and input. In this paper, when we mention FSA, we mean Deterministic FSA.

Execution of an experiment causes the FSA to move from state to state via transitions. Each state has a widget list indicating status of experiment widgets in this state, and a table for possible transitions starting from this state. Once a transition is defined, its information is stored in the transition table of its starting state. Information in the transition table includes the name of the transition, required interaction with a particular widget from the user (e.g. press a button) for the traversal of the transition, what is necessary in order to move from one state to another (i.e. the condition for the transition), and certain manipulation of the model as the result of the transition (i.e. the consequence of the transition). The table also provides the next state the FSA has to go to as a result of the transition.

Expressions consist of experiment variables, operators, numbers, and functions. There are two types of expressions in CyberLab experiment models: Boolean expressions are often used as transition conditions; assignment expressions are mostly used as transition consequences. There are also two types of experiment variables: dependent and independent variables. A dependent variable is associated with a function consisting of various operators and references to other independent or dependent variables. An independent variable has an initial value and its value is independent of other variables.

In order to determine whether a transition condition holds, we need to design a mechanism to evaluate values of expressions and variables. For an assignment expression, the evaluation process will first parse the right hand side of the expression. If the right hand side contains a variable, the current value of the variable will be retrieved from the experiment model. The evaluated value of the right hand side will be assigned to the left hand side variable, and the new value of the left hand side variable is stored back into the experiment model.

Boolean expressions are evaluated in a different way. The left hand side of a Boolean expression may be a combination of variables, operators, and numbers, while the left hand side of an assignment expression usually consists of only one variable. Another difference is that relational operators, instead of assignment operators, are used to link the left hand side and the right hand side. As a result, both left and right hand sides of Boolean expressions are evaluated. Their values are then compared based on the relational operator. A Boolean expression is evaluated to integers. Zero represents False, while other values are interpreted as True. No variable's value is going to be changed after the evaluation of a Boolean expression.

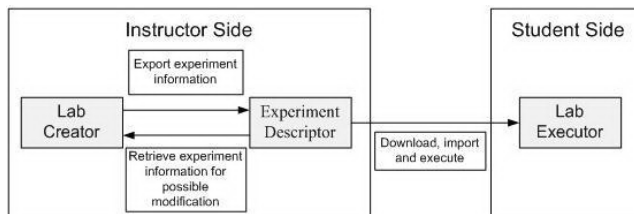


Figure 1. Architecture of the CyberLab toolkit.

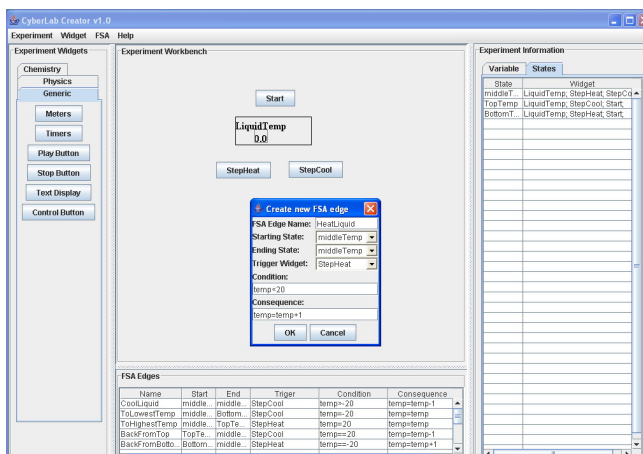


Figure 2. Graphical user interface of LabCreator.

### 3. Architecture of the CyberLab toolkit

As shown in Figure 1, the CyberLab toolkit consists of three major components: LabCreator, Experiment Descriptor, and LabExecutor.

LabCreator, which is written in Java, is designed to be a “what you see is what you get” virtual experiment editor and supports the “drag and drop” feature for positioning experiment widgets. It also provides a library of experiment widgets, such as buttons, text display, tubes. In order to create a virtual experiment, instructors of online science courses first use LabCreator to choose what widgets to use, and design what the virtual experiment will look like, including the layout and the label of experiment widgets. How the experiment functions is specified by creating the corresponding experiment model and properly setting up FSA, variables, and expressions. Usually, it is also necessary to specify a correlation between the various visual components of the experiment with the variables of the underlying model so that users can visually observe changes in the ongoing experiment. For example, one has to associate the color of fluid within a beaker with the variable

that denotes the temperature of the fluid. The creation and specification of experiment variables and FSA can be done through pop-up dialog windows.

Figure 2 is the graphical user interface of LabCreator, within which a simple fluid heating and cooling experiment is being designed. Several widgets have been labeled and positioned on the “Experiment workbench”. Existing FSA states and experiment variables are shown in the “Experiment Information” panel to the right of the “Experiment Workbench”. The pop-up window is where users design FSA transition edges. Users can specify the edge’s name, starting and ending state, the widget that will trigger the transition of the edge, and condition and consequences of the transition of the edge. Information about existing FSA edges can be found at the “FSA Edge” panel underneath the “Experiment Workbench”.

Therefore, all instructors have to do basically include designing FSA to represent the experiment, pressing mouse buttons, dragging and dropping experiment widgets, and typing texts and equations in pop-up dialog windows. Since no programming is needed to use LabCreator, those who do not know programming languages are enabled to create their own virtual experiments in an intuitive and easy way.

During the design process, the experiment can also be previewed within LabCreator. If the instructor is satisfied with the experiment, he or she can export the experiment into “experiment descriptor”, a file that contains the definition and description of the experiment. These files are generated by serializing experiment objects with Java Object Serialization technology. The descriptor files will be stored on the instructor’s web server of online courses.

LabExecutor is used by students to conduct virtual experiments. In LabExecutor, students will get the experiment with the same functionality and appearance as designed by instructors in LabCreator. LabExecutor is a Java Applet, which can be executed in most web browsers. To run an online experiment, students need to visit the instructor’s website for the online course through web browsers and invoke the LabExecutor. LabExecutor then downloads the experiment descriptor file from the instructor’s web server and de-serializes experiment object. The de-serialized experiment object will be used to determine the functionality and design of the experiment. Finally, the experiment is displayed in web browsers, and students are allowed to play with the experiment.

After the implementation of the toolkit, several tests with simple experiments were conducted and CyberLab was able to work as we planned.

## 4. Conclusion

In this paper, we presented the CyberLab toolkit, which provides a package of services for online virtual laborato-

ries, including virtual experiments creation, preview, and export at the instructor side, and experiments loading and execution at the student side.

We consider this toolkit to be an important progress with unique characteristics in the research of online virtual laboratories. To our knowledge, CyberLab is the first toolkit that does not require creators of virtual experiments to know programming. By freeing its users from learning programming and coding, CyberLab may expand its user population among instructors of online science courses, which in turns facilitates the popularization of virtual experiments in online science education.

Admittedly, there is still a long way to go before CyberLab can be widely used by the public. In the future, we need to enrich the experiment widget library of CyberLab so that the toolkit can be used to create more experiments for different disciplines. The number and variety of experiment widgets supported by the toolkit will be very important for the success of CyberLab. In addition, we plan to improve the user interface, making it easier and more intuitive for users to create experiment models, and to add data collection feature.

## References

- [1] W. Christian and M. Belloni. *Physlets: Teaching Physics with Interactive Curricular Material*. Prentice Hall, New York, NY, 2001.
- [2] U. Harms. Virtual and remote labs in physics education. In *Proceedings of the Second European Conference on Physics Teaching in Engineering Education*, Budapest, Romania., 2000.
- [3] J. Kim, S. Park, H. Lee, K. Yuk, and H. Lee. Virtual reality simulations in physics education. *Interactive Multimedia Electronic Journal of Computer-Enhanced Learning*, (February), 2001.
- [4] Y. Lee, W. Ma, D. Du, and J. Schnepf. Creating a virtual network laboratory. In *Proceedings of the International Conference on Multimedia Computing and Systems (ICMCS'97)*, pages 642–643, Ottawa, Ontario, Canada, 1997.
- [5] M. Morozov, A. Tanakov, A. Gerasimov, D. Bystrov, and E. Cvirco. Virtual chemistry laboratory for school education. In *Proceedings of the 4th IEEE International Conference on Advanced Learning Technologies, (ICALT'04)*, pages 605–608, Joensuu, Finland, 2004.
- [6] R. Subramanian and I. Marsic. Vibe: Virtual biology experiments. In *Proceedings of the tenth international conference on World Wide Web (WWW'01)*, pages 316–325, Hong Kong, 2001.
- [7] K. Zhao. Implementation of labcreator and the integration of cyberlab. Master’s thesis, Eastern Michigan University, Ypsilanti, MI 48197, USA.