The Reactor Lab

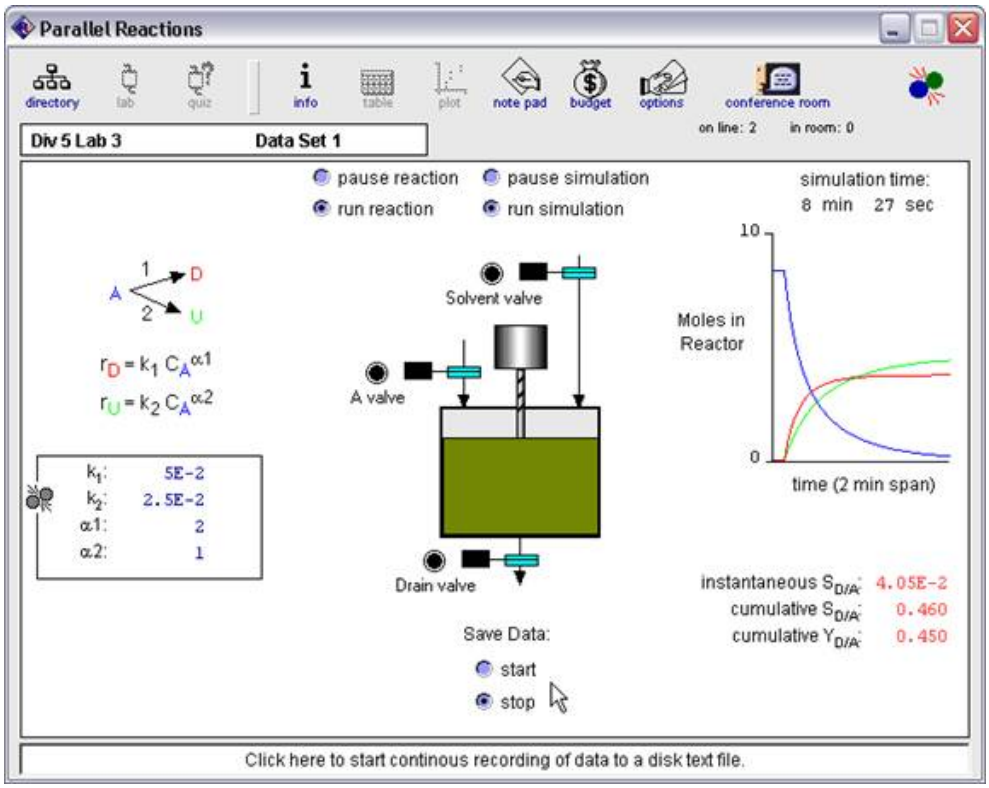
Richard K. Herz <herz at ucsd.edu>, University of California, San Diego, California 92093-0411, USA.

The Reactor Lab provides a framework for distribution of learning modules, specifically interactive simulations. In the current jargon, the Lab can be considered a "desktop rich internet application." Although not essential to revJournal readers, the Lab's modules happen to be simulations of chemical reactors.

Soon after I started teaching, I learned that teaching by lecture and textbook alone did not satisfy students' needs. Many systems are too complex to be conveyed fully by the static plots and drawings in books. Essential to learning is active practice and application of new knowledge. Real experiments are wonderful - but cost and space constraints limit the number which can be implemented - usually from zero to a few in most courses. And so I set out to write software simulations of lab experiments.

A student obtains the Lab by downloading a distribution from the web. The distribution includes a Revolution standalone application and associated stacks. On startup, the Lab opens to a "Directory" in which the student can select a "division" of related modules, view the list of modules or "labs" in that division, and open one of the modules.

Figure 1 shows two reactions competing with each other. The contents of the reactor are plotted continuously on the right side of the window at a rate that is controlled by the script to be proportional to simulation time (1 min of simulation time in 10 s real time). No animals are harmed in this experiment: the "mole" on the plot is a standard chemical measure of number of molecules! The student can change reaction inputs, open and close valves, pause and resume the reaction and the simulation, and save data to disk files.



A few modules are supplied with the initial distribution. To get others, the student clicks a button which connects the Lab to the Internet. When a module is selected, the main stack and associated files are downloaded and installed in a folder on the client disk. When the student is off line, they can use previously accessed modules. This is one of the advantages of using Revolution rather than deploying modules in web pages.

On the first connection in a session, an updater stack compares a list of names and dates of library stacks on the client with the corresponding list on the server, and then downloads any stacks which need updating. The stack which is made into the standalone executable contains only a couple lines of script which get the path to the executable on the client disk (the root path to all other files) and then call a "bootstrap" handler in a library stack to get things started. This "bare standalone" structure allows all the other scripts and stacks to be updated over the Internet. Most stacks, with the exception of the Internet library (libUrl) and the small stack that downloads the updater stack, can be updated in the current session by deleting the old version from memory and opening the new version. New versions of the two exceptions become active in the next session.

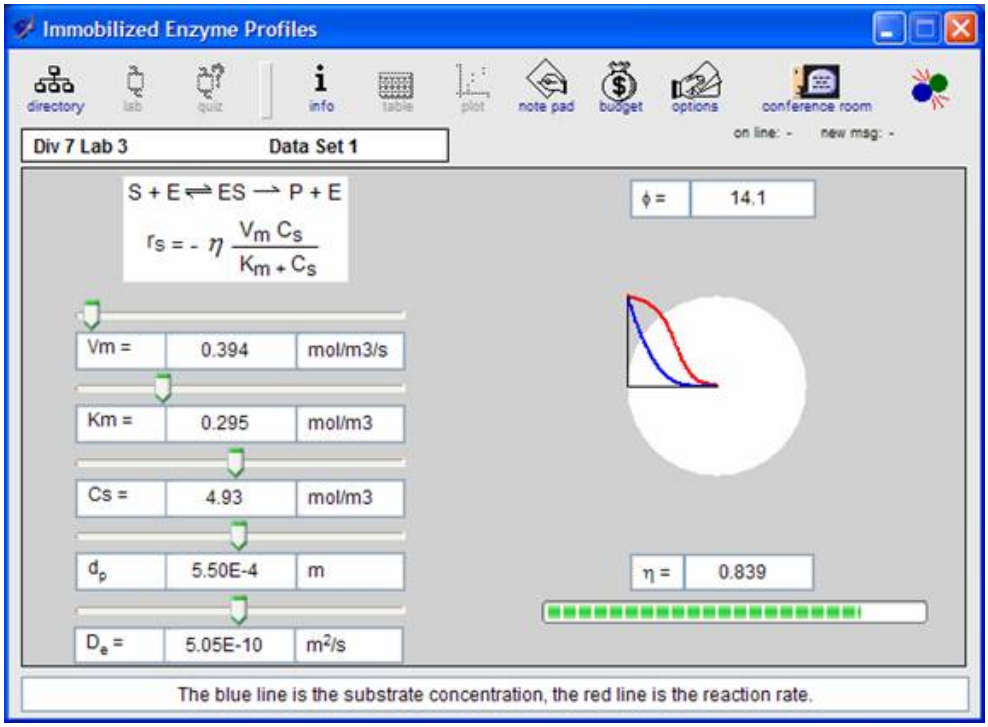
Subsequent times a student goes on line, the dates of local modules are compared to dates on the server and local labs that are old are labeled as such and are updated when next accessed on line. The folder structure of the Lab on the client mirrors the folder structure of the lab on the server. Plain text files are used for the lists of file names and dates on the server and clients.

The automated update features fit well with the limited time I have available to work on the project. I don't have time to schedule and build regular revisions and then notify users and distribute new versions to them. Now, whenever I make a change or develop a new module, I just upload it to the server and it gets to users automatically when they need it. This update model is becoming a standard method of maintaining users' software.

The Lab supplies the experiment, a way to change conditions, run experiments, and save data to disk files. Using Revolution stacks rather than a web page allows students to save experimental data to disk files, just as they would if they were taking data in a real lab. Then they can use standard software to analyze their data. Students can collaborate on homework using the Conference Room, which receives messages at a server and broadcasts them using Rev socket commands.

Many of the modules have two sections: the main entry point in which all inputs and outputs are known to the student, and a quiz section. In the quiz section, the student must run experiments, analyze their data, and then check to see if they have an answer within an acceptable range. The Lab charges virtual $ for each experiment to teach students that they should not run experiments indiscriminately, and then awards them virtual $ for correct answers. The Budget Report records the history of each quiz. Students must turn in their data, analysis work, and a copy of the Budget Report which contains an authentication code.

In most cases, Revolution script is sufficiently fast for the mathematical computations required by our simulations. For example, Figure 2 shows a module that continuously updates plots of substrate (reactant) concentration and reaction rate within a porous particle of immobilized enzyme as the student drags an input scroll bar. In this simulation, Rev script repetitively solves a second-order ordinary differential equation using the iterative shooting method as the scrollbar is dragged. In Figure 2, S is substrate, E is enzyme, P is product, the inputs Vm and Km are reaction rate parameters, Cs is the substrate concentration over the external surface of the particle (white disk), dp is the diameter of the particle, and De is the diffusion coefficient.



Cases where Rev script isn't fast enough are where large numeric arrays are involved. In such cases, we couple Rev to externals which are written in a language designed to handle large numeric arrays, such as C or Fortran. The advantage to doing as much as possible in Revolution, of course, is that Rev scripts are cross-platform, whereas externals have to be compiled separately for each platform.

The Lab does not provide tutorials on the background materials. Thus, the Lab must be used in conjunction with a textbook. Since there are excellent textbooks in the field, I chose to devote my efforts to developing something missing from the field. However, the lack of supporting explanations and complete learning modules has probably limited use of the Lab by other educators.

A comment on software usage by instructors: In my cynical view, most instructors are too comfortable with the model of requiring a textbook, assigning end-of-chapter problems, and having graders score the work using a solution manual provided by the textbook author. Incorporating anything outside this model has a huge "activation energy," in the language of a chemist, and is unlikely. This was more true when students didn't own computers and instructors had to exert significant effort to get software budgeted, licensed, and installed in a campus computer lab. Fortunately the barriers are lowering now that most students own their own computer. Now, it is easy to give students a web URL in an assignment, even to ask the student to pay for software on the web, although the instructor still has to make the effort to incorporate the software into their course. My ideal, given time and funding, would be to provide complete learning modules - explanations plus labs - that would be easy for an instructor to integrate into a course, or even serve as the complete course material.

I'd like to give you a brief history of the Lab in order to make a point pertinent to the advantages of working in Revolution.

The earliest version of the Lab was developed as a single HyperCard stack and used in my course in 1993. In 1996, in order to facilitate addition of new modules, the Lab was split into separate stacks, with each module in a separate folder. This structure required the significant challenge of learning AppleScript, as a note to those accustomed to the built-in folder operations of Revolution. The Lab was posted on the web and made available to other educators in 1996. In 1999, the Lab was converted from HyperCard to MetaCard, which allowed making a Windows as well as a Mac version. Ninety five-plus percent of current downloads are Windows versions, although I insist on providing at least a Mac version as a matter of principle.

Sivakatirswami's Himalayan Academy Cyber Study Hall, which started deploying MetaCard stacks via the Internet in 2000, and Richard Gaskin's 2002 RevNet, served as inspirations for the Lab's Internet version. In 2003, the Lab was

converted from MetaCard to Revolution, and the first version with automatic script update and module distribution via the Internet was posted.

The main point of the history is the continuity of the scripting language and the stack format over 14-plus years. Even though the steward of the language has changed from Apple, to Claris, to Apple, to MetaCard, to Runtime Revolution, relatively few modifications to legacy script was required. Some of the scripts from the first 1993 modules are present in the current Lab.

This continuity is a crucial reason that I have been able to continue development of the software as a side-line endeavor. I doubt that my efforts could have survived the time commitments of changing languages or restructuring the software drastically. There's no guarantee of the future of Revolution but the history of the language and the increasing rate of development of the environment bode well.

The lab is distributed free of charge via the web at www.reactorlab.net. Over 16,000 people have downloaded the Lab from 103 countries in the last 6 years (through January 2006). We have heard from students and instructors using the Lab in courses in Bolivia, Brazil, Chile, Italy, Mexico, The Netherlands, South Africa, Spain, Taiwan, Turkey, and the U.S.

Most of our current efforts are devoted to a new project at www.purewaterlab.org, which uses the framework of the Reactor Lab in a different field. We are collaborating with people at the University of Arizona who are developing the explanatory text and graphics. We are using Revolution's embedded browser technology (formerly Altuit's altBrowser2) to accommodate their desire to have more formatting options than standard Rev fields allow. Both projects have received support from the National Science Foundation.

I would like to thank my fellow scripters for their help through the email lists, and also thank Scott Raney of MetaCard and the staff of Revolution for their development of this wonderful - amazing - environment.