Overview

We believe that physics students need programming skills to succeed in modern research and development laboratories. Physics 111 students currently learn some programming in LabVIEW and Matlab, two standard scientific applications that are used in the 111 lab to acquire, process, and analyze data. Since May, 2007, we have been experimenting with the use of a general purpose development environment, C# with .NET or Measurement Studio, to interface computers with our experiments. We have created a new application called Cuttlefish consisting of a lightweight framework and stand-alone modules in C# and .NET to control experiments and process data. We expect this approach to allow scientists who are not full-time programmers to take advantage of the power and speed of this modern general-purpose
We are partnering with the MIT Biological Engineering Department [1], whose undergraduate lab course 20.309, Biological Engineering II: Instrumentation and Measurement [2], resembles our Physics 111 lab in scope and intensity. Our shared interest in biophysics experiments is the context of our collaboration. At Berkeley, we replicated a optical trap experiment [3] designed by David Appleyard at MIT, but developed new software in C# and .NET to control the experiment and take data. We also revised our particle tracking experiment, Brownian Motion in Cells [4] to run in C# instead of Matlab.

We are grateful for the generous support of the Microsoft Research Division to make this project possible.

People

U.C. Berkeley Physics Department

- **Jan Liphardt** - Assistant professor and resident biophysicist.
- **Tom Colton** - Coordinator and keeper of the budget.
- **Suneet Upadhyay** - Undergraduate student, programmer, and optical trap builder. (2007-2008)
- **Matthew Rocklin** - Staff research associate, programmer, and particle tracker. (2007-2008)
- **Trevor Owens** - Staff research associate, developer of optical trapping experiment. (2008)
- **Don Orlando** - Physics 111 lab director, apparatus guru.
- **Jim Siegrist** - Professor and Physics 111 Lab Committee Chair.
- **Johnny Phan** - Undergraduate student researcher. (2008)
- **Jon Aytac** - (summer 2007) Graduate student and programmer.

MIT Biological Engineering Department
Steve Wasserman - 20.309 course instructor, programmer extraordinaire.

Software

We developed a lightweight framework for configuring and connecting scientific computing functions so that users can easily create, combine, and re-use modules to solve real problems in the lab. The framework is written in C# with Microsoft Visual Studio. Modules have been developed at Berkeley for two biophysics experiments, Brownian Motion in Cells [4] and Optical Trapping [5]. At MIT, the particle tracker has been modified for their instrumentation lab and a new module was developed for their atomic force microscope.

Using the framework, we can modularize the software component of our experiments. For example, we create modules that control hardware devices and expose their settings for real-time modification, modules that read or write data on experiment hardware, and modules that manipulate and analyze data. Each module has a documented set of input events and output events, and these events are connected to form a functional program. User interface elements are also modularized and they support input events and output events. An event made by one module and an event listener in another module constitute a connection between the modules. Each event sends a custom EventArgument between the modules, such that many objects of arbitrary type can be moved between modules with each event that is sent. Such general connections between modules allow users to separate and package software functions however it is appropriate, since arbitrary information can be moved from point to point.

Our framework uses an XML schema to instantiate the modules and specify the topology of the program. Some of the instantiated modules are user interface elements, and these are connected to modules that will utilize the input. We commonly use a linked-chain...
topology whereby a module utilizing the system timer or else a hardware timer is connected to a module that acquires some data, such as an image frame or 1 second of DAC measurements. Then, the module that acquired data sends a frame or packet of data to a chain of modules that perform an analysis and present a result. Since the modules are arranged using XML files, the user can adjust the analysis chain by modifying the XML file to use different modules or to use modules in a different order.

Given a laboratory utilizing disparate computer-hardware interfaces and software analysis requirements, the framework can be used to produce a system of compatible, interchangeable modules to acquire and analyze data. Since it is based on the powerful .NET framework, our system can be used to control any device with drivers for the Windows platform. Most computer controlled hardware used in our lab is compatible with this platform. Given a set of compatibly designed modules for available hardware, and a compatible set of analysis routines, it is possible for users to modify existing experiments or to build new ones without writing any C# code. Also, should there be a need to utilize new hardware or analysis routines, users can create new modules in any .NET language to implement the necessary functions. Then, the new modules can be used with the library of pre-existing ones to minimize the amount of re-development required for a new experiment.

Download the current Framework 1.05 as a zip file [6].

Powered by Drupal

Source URL: https://experimentationlab.berkeley.edu/netproject

Links
[4] https://experimentationlab.berkeley.edu/BMC
[5] https://experimentationlab.berkeley.edu/OTZ