Biophysical Journal Volume 69 November 1995 2168-2173

Teaching Computer Interfacing with Virtual Instruments in an Object-Oriented Language

Miriam Gulotta
Department of Biochemistry, University of Minnesota, St. Paul MN 55108 USA

ABSTRACT LabVIEW is a graphic object-oriented computer language developed to facilitate hardware/software communication. LabVIEW is a complete computer language that can be used like Basic, FORTRAN, or C. In LabVIEW one creates virtual instruments that aesthetically look like real instruments but are controlled by sophisticated computer programs. There are several levels of data acquisition VIs that make it easy to control data flow, and many signal processing and analysis algorithms come with the software as premade VIs. In the classroom, the similarity between virtual and real instruments helps students understand how information is passed between the computer and attached instruments. The software may be used in the absence of hardware so that students can work at home as well as in the classroom. This article demonstrates how LabVIEW can be used to control data flow between computers and instruments, points out important features for signal processing and analysis, and shows how virtual instruments may be used in place of physical instrumentation. Applications of LabVIEW to the teaching laboratory are also discussed, and a plausible course outline is given.

INTRODUCTION

The development of powerful personal computers and workstations has transformed the way biophysicists and other scientists work. Increased processing speed and available memory have led to the development of highly sophisticated programs that perform intricate calculations and handle large data sets. Electronic signal processing can now often be replaced by digital computer processing. Windows and graphical user interfaces have made it possible for computers to perform multiple tasks simultaneously and have made it easier for scientists to analyze and display data as well as to write papers, manage references, and so on.

Although computer interfacing software has taken advantage of faster processing, development of user-friendly languages for integrating machines and computers has been slow. Integration of computer interfacing software into the curriculum has been slower still. The purpose of this article is to describe a powerful, widely applicable approach to interfacing that uses object-oriented programming and the concept of "virtual instruments" (VIs) and to show how this approach can be used in the classroom and the research laboratory.

This discussion focuses on LabVIEW, a computer language developed by National Instruments for computer interfacing through their data acquisition boards, which are purchased separately. National Instruments is not the only company to take this route: DEC's Realtime Integrator, Sparrow's KMAX, and Hyperception's Hypersignal-Windows AMPS are other examples. However, LabVIEW is probably the most commonly used software of this type, is available for a wide variety of computer systems, is a complete language, and is user friendly.

The basic concepts described here should be applicable to the other software packages.

COMPARISON WITH OTHER INTERFACING METHODS AND LANGUAGES

Many instrument manufacturers have developed their own interfacing languages, which are designed to be more user friendly and more easily modified by the casual user than are standard languages like Assembler and C. Most modern instruments of any sophistication probably have such packages. There are two major problems with this: every company has its own language, and the user is generally only given enough information to modify prewritten routines and not enough information to write full-scale programs. In general, these programs save their data files in a form that is not recognized by standard data processing and graphing programs, but there is usually a way to save the data in ASCII format. Although the prewritten software may be easy to use, the language it is written in is usually not user friendly, and the accompanying manuals often give little information, if any, about using the language for programming.

Users can also write their own interfacing routines in any standard programming language from Assembler to C. The user buys a data acquisition board that can handle all of the input/output signals and then writes a program to direct the flow of information over a bus on the data acquisition board. The user must know a considerable amount about the computer language and the bus architecture and about routines to send/receive data over a bus. The learning curve for doing this is steep, but once the method is understood, future programs are easy to create.

Working with LabVIEW is very different from working with languages like Basic, FORTRAN, and C, which are

Received for publication 8 August 1995 in final form 8 August 1995.

Address reprint requests to Dr. Miriam Gulotta, 140 Gortner Laboratory, University of Minnesota, 1479 Gortner Avenue, St. Paul, MN 55108-5009. Tel.: 612-624-7468; Fax: 612-625-5780; E-mail: gulotta@molbio.cbs.umn.edu.

0006-3495/95/11/2168/00 $2.00

text-based and procedural. LabVIEW uses a graphic, object-oriented computer language. This type of language takes advantage of the Macintosh or Windows-type operating system, and every variable, function, or subroutine is represented as an object. An object may be placed anywhere on the screen; its precise location is unimportant. Data flow between objects through "wires." To follow the logic of a LabVIEW program, you follow where the data go, not the order of commands.

In LabVIEW, instead of writing programs you create VIs. VIs have front panels just like any other instrument on your bench. The input variables are set the same way as the settings on a "real" instrument, i.e., by turning knobs and flipping switches. Output is displayed using LEDs, charts, graphs, digital indicators, etc. At the heart of a VI is a sophisticated computer program that attaches functions to the controls and controls data flow between the computer and the objects hooked into the bus. VIs may be run as autonomous programs or may be incorporated into other programs. A VI that is called from within another VI is a subVI; it is the equivalent of a subroutine in a text-based program.

LabVIEW is designed for use in the laboratory. The features that make LabVIEW stand out as a research tool are the ease of controlling data flow between computers and instruments, the large number of algorithms that come with the software as premade VIs that can be easily incorporated into other VIs, the ability to incorporate code from other programs into LabVIEW VIs, and the ability to make VIs that may be used in place of physical instrumentation.

CONTROLLING DATA FLOW BETWEEN COMPUTERS AND INSTRUMENTS

As an example, consider using LabVIEW to run several digitally controlled devices that are all attached to digital port 0. You might use a VI like that shown in Fig. 1 to send messages to the devices. The top frame of Fig. 1 shows the icon of the VI. The icon allows information to flow between VIs, as will be demonstrated later on. The middle frame of Fig. 1 shows the front panel for this VI. On it are an on/off switch, a string control labeled "port," and a dial that contains all of the possible messages that could be sent. These controls may be set by using the index finger tool. The VI may be executed from the front panel. The third frame is the block diagram of this VI, the actual program code. The top large rectangle with the patterned border in the block diagram labeled "true" is part of a Case structure. This is the LabVIEW equivalent of a Case statement. Wired to the question mark on the left side of the rectangle is the terminal of the on switch. When the switch is on, the output from its terminal will be "true," and the program will execute the code within the true rectangle of the Case statement. If the switch is off, the terminal value will be ''false'' and the code in the false rectangle, shown directly under the true rectangle, will be executed. Within the true rectangle are three smaller rectangles: one labeled DIG PORT, one containing the number 4, and one containing the letters DBL and labeled message. DIG PORT is a subVI that is used to send digital messages over the bus. The 4 is a locally defined constant that is a required input for the DIG PORT VI. The rectangle labeled "message" is the terminal of the message dial shown on the front panel. The additional wire from DIG PORT VI is wired into the left-hand side of the case statement. This connection is a tunnel that allows information from outside the case structure to be used by the VIs inside. Once a tunnel is created, the transferred data may be used for any or all cases. The terminal of the front panel string control, labeled "port," is wired to this tunnel. It is desirable to leave string control outside the case structure because it is needed regardless of which case is chosen, and its value is not case dependent. When the true case is executed, the message determined by the front panel dial is sent across digital port 0. Execution of the false case results in the message 0 being sent.

The Write to Dig Port. vi is the simplest way to send information to digital instruments. It is a premade VI and is accessible through the Functions menu of the diagram window. There are three levels of premade VIs that may be used for digital information. The higher level controls allow the programmer more flexibility and allow for handshaking, but they are more complex. The Write to Dig Port. vi is actually composed of higher level digital VIs.

Working with analog ports is similar (Fig. 2). Suppose you want to read the voltages from a photomultiplier tube and store this data in a file. Fig. 2 is a VI that uses the simplest analog input VI to monitor the signal from channel 0, plots it versus time, and stores the data to a spreadsheet file. On the front panel are controls that allow the user to input the channel number, the number of samples to be acquired, the rate at which to collect the data points, and the name of the file where the data are to be stored. The diagram for this program is extremely simple. The rectangle labeled "AI Acquire Waveform.vi" is a VI that comes with the software. It is composed of advanced VIs that configure the analog port specified, allot memory for data acquisition, set the scan rate, and acquire the given number of data points. The data are stored in a data array that can be used by subsequent VIs and functions. In addition, the actual time period between data points is calculated. The data array is carried through a wire (the thick line protruding from the right side of the rectangle) to a function labeled "bundle." The data array, the initial starting time, and the time/pt are bundled together, and the bundle is then transferred to the graph on the front panel. The resulting graph will be intensity versus time, where time increases one period for each data point. The data array is also transferred to another VI, which is labeled "Write To Spreadsheet File.vi." This VI puts the data into spreadsheet format and saves them to the file indicated by the front panel control file path.

Notice that there are different types of wires. The type of wire used is indicative of the data type of the variable being transferred; thick solid lines represent 1-D arrays, thin solid

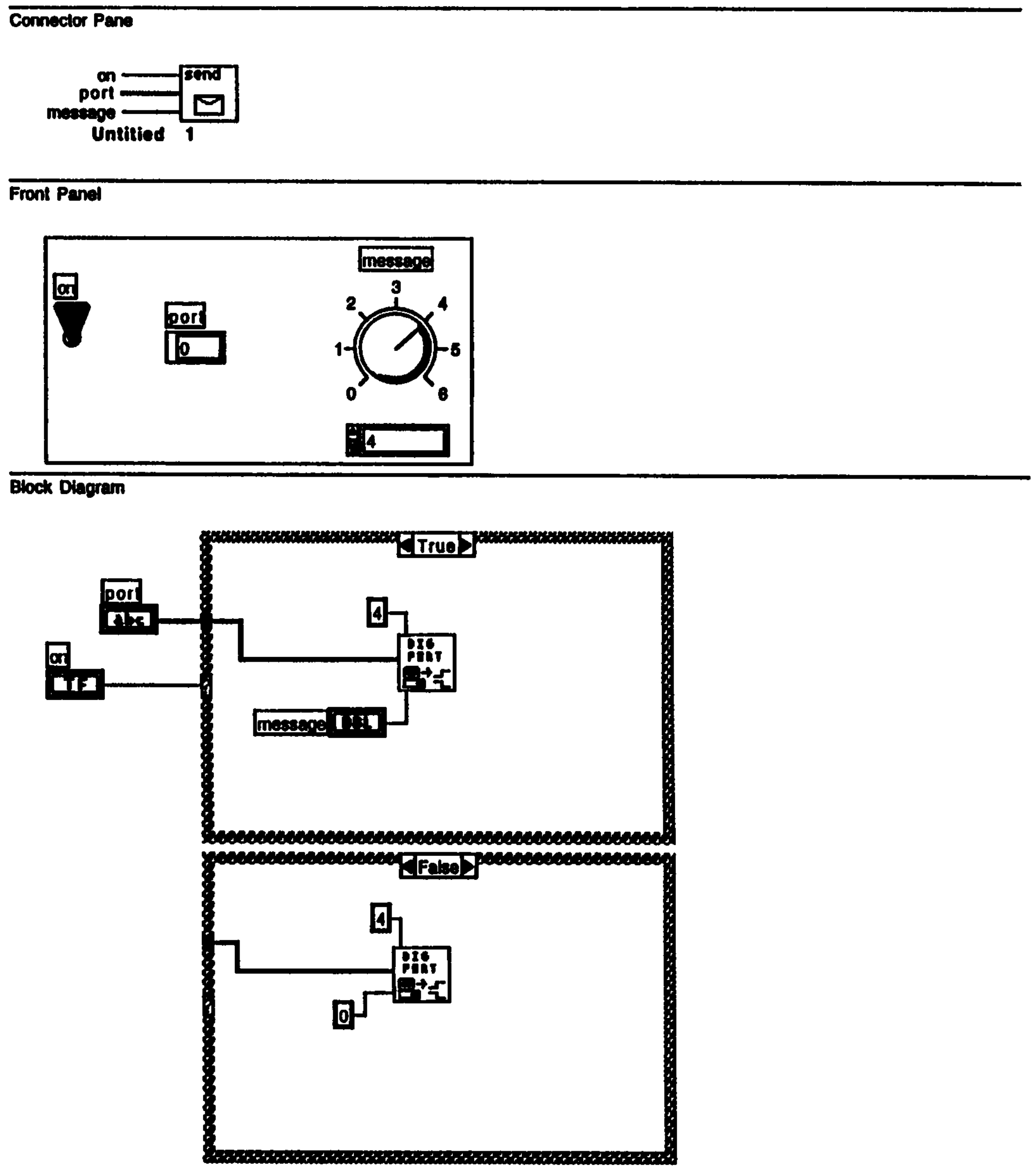

FIGURE 1 Digital message-sending VI. When the front panel on-switch is on, the message indicated on the dial is sent to digital port 0. When the on-switch is in the off position, the message 0 is sent. The icon shown on the connector panel allows the front panel controls to be set from another VI.

lines represent numbers. The data array is transferred to two different places: the bundle function and the Write to Spreadsheet VI. To the programmer, these two tasks occur simultaneously. In the actual execution of the VI, LabVIEW executes VIs one at a time using an operation hierarchy.

In the above examples, information was passed between the user and the diagram window by means of the front panel controls. Transfer of information between two VIs is done through their icon connectors. In Figs. 1 and 2, the rectangles labeled "DIG-PORT," "AI Acquire Waveform.vi," and "Write to Spreadsheet" are all VIs themselves. The picture shown is the icon. Beneath the icon are a group of terminals that are wired into the controls on the front panel. The analog data array was transferred from the AI Acquire Waveform.vi VI through an output terminal of the AI Acquire Waveform.vi icon. The thick wire carries the data to the appropriate input terminal of the icon of the Write to Spreadsheet VI. The icons for the VIs created in Figs. 1 and 2 are shown as the top segment of each figure. In both cases all of the front panel controls have been wired to the icon. Input controls are often connected to terminals on the left-hand side of the icon, and output controls are wired to the right. This is just a convention and not a requirement. Only the controls on the front panel that must to be passed between VIs need to be wired to the icon; however, a variable that was assigned in the diagram of a VI may be passed out of that VI only if it is wired into a front panel control that is wired to the icon of the VI.

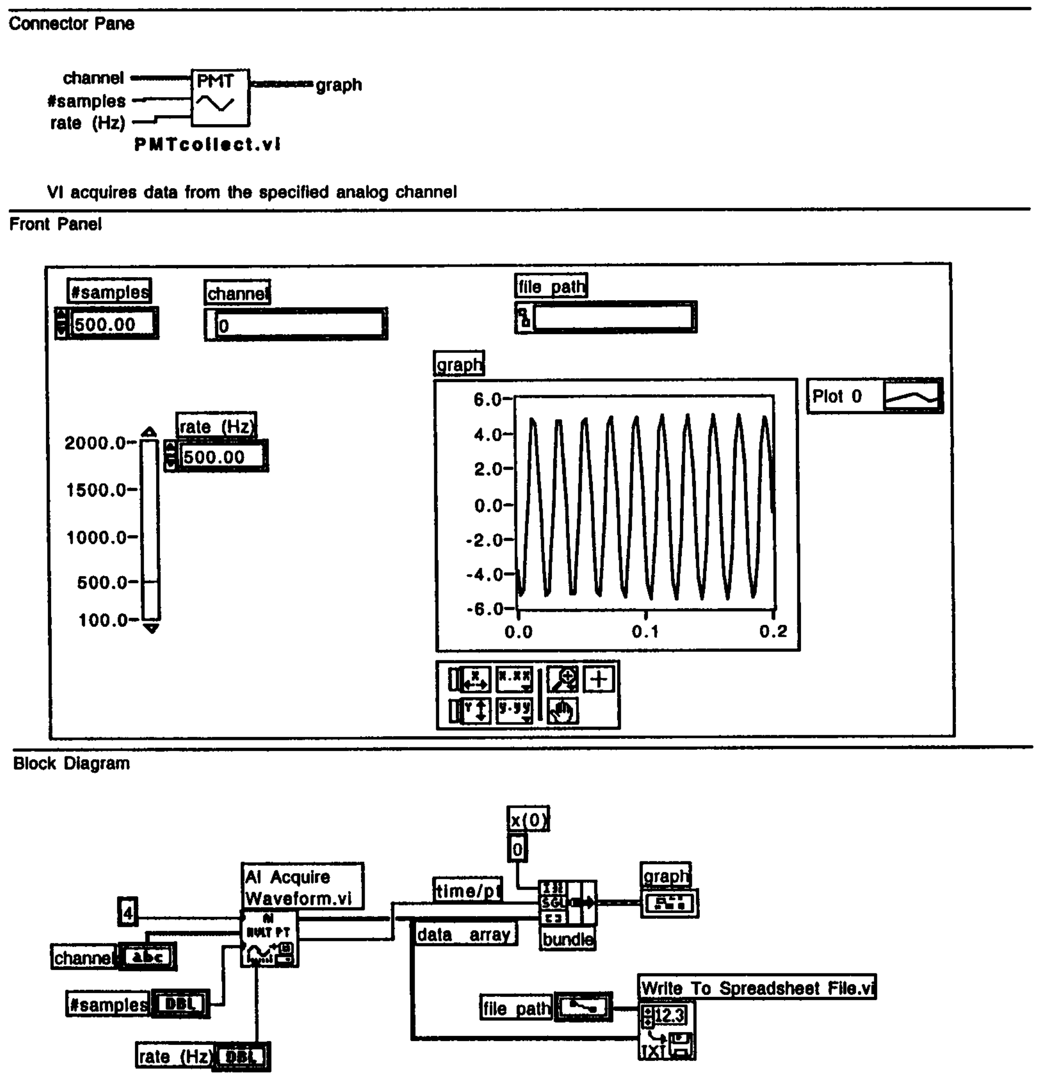

FIGURE 2 Analog-input VI. The number of samples to take, the data acquisition frequency, and the channel to be read are all determined from front panel controls. The acquired data are displayed on the front panel graph and saved to a file. The icon shown on the connector panel allows remote setting of front panel controls and makes the acquired data available for use by subsequent VIs.

OTHER IMPORTANT FEATURES

Premade VIs

LabVIEW provides a large number of prewritten analysis VIs, greatly simplifying the work of the scientist. The algorithms most used for processing data are all provided with the software. The algorithms that are used are given in the manual, but each VI works like a black box. The raw data source is wired into the inputs of the VI, and the finished, processed product is available from the output terminals.

Signal processing VIs include fast Fourier transforms, power spectral analysis, Hilbert transforms, and many more. Time-series analysis VIs include convolution, deconvolution, various correlation VIs, derivative and integral analysis, etc. There are also ready-made finite and infinite impulse response filters as well as nonlinear filters, and 12 different smoothing windows, a wide variety of statistics VIs, regression VIs, linear algebra VIs, and VIs that perform array functions. Also, premade VIs allow you to take advantage of the timing chips on the National Instruments data acquisition boards. The set of available analysis VIs has expanded greatly with each software update, and this is likely to continue.

Incorporating code from other programs

Apple Events VIs allow the execution of other applications from within LabVIEW VIs. For highest performance applications, C-code can be incorporated using a code interface node (CIN). Many of the analysis VIs that come with LabVIEW use CIN nodes. For very high level mathematics and the ability to make contour plots in three dimensions, the numerical mathematics program HiQ can be directly accessed by LabVIEW using an HiQ interface, which must be purchased separately. Recently, National Instruments announced that Microsoft Excel will be accessible through LabVIEW.

USING VIs IN PLACE OF PHYSICAL INSTRUMENTS

The set of resources available through LabVIEW gives the user the capability to build VIs that replace physical instruments. Essentially, any instrument having functions that can be digitally performed may be replaced with a VI. The limitations of the VI will depend on the limitations of the data acquisition board(s) as well as on the processing limitations of the controlling computer. National Instruments provides several examples of this with the software. There is a premade VI that acts like a 40-kHz oscilloscope, and a signal generation VI that produces sine waves, saw tooth waves, etc. Any of the premade VIs may be altered and used as subVIs in other VIs.

The data acquisition boards are expensive, so building a VI may cost as much as buying a physical instrument. However, the boards are not limited to one application, so one board may be part of several different VIs. In addition, VIs are much easier to modify and update than physical instruments. VIs may send/receive information to/from any other instrument on the bus.

APPLICATIONS IN THE TEACHING LABORATORY

There are at least two ways in which LabVIEW may be used to make improvements in the classroom. One is to update old instruments or build VIs where physical instruments are lacking. The other is to teach students about object-oriented programming and computer interfacing.

In this age of quickly advancing technology and ever-diminishing funding, LabVIEW offers a good solution for keeping teaching labs up to date. A well-made older instrument that lacks the ability to do sophisticated signal processing may be brought up to date by feeding its signal into a new data acquisition board and using subVIs to process the data. This is not a trivial task, and it requires a significant amount of labor; however, it is good way to keep the instruments up to date, and VIs do not have parts that will decay over time.

LabVIEW may also be used to teach students about instrumentation. As more and more functions are computer controlled, the amount of knowledge required to run an instrument decreases. Consequently, the instrument becomes a black box with input and output ports; the student does not have to know how the instrument actually works. Faculty members willing to teach courses in instrumentation are not always available. LabVIEW is relatively easy to use, and a wide selection of virtual instruments can be created. VIs may be linked together, and the output of one VI can be used as the input for the next VI. For example, the waveform generated by the Waveform Generator VI can be used as the input signal for the 40-kHz Oscilloscope VI. Students may also design their own instruments with a minimum amount of hardware required.

There are several universities and one or two high schools that base courses on LabVIEW. To encourage this classroom use, National Instruments in conjunction with Prentice-Hall Publishing has produced a student edition of LabVIEW. The manual is set up to be used as a course text. It is detailed and assumes no prior knowledge of programming or of sending information over a data bus. An instructor's manual is also available.

The student edition has most of the features of the regular edition, can be used to drive data acquisition boards, and costs about as much as a standard textbook. The drawback is that many prewritten complex VIs cannot be opened or modified. Programs that are written using the student version may be read by the standard software, but the reverse is not necessarily true. However, VIs can be opened and modified regardless of the presence of a data acquisition board, so students can work with LabVIEW on their home computers.

DESIGNING A COURSE BASED ON LabVIEW

A course on laboratory instrumentation for biophysics students and those in related sciences might proceed as follows.

1) Start with Chapter 2 of the manual, "LabVIEW and Data Acquisition, Transfer, and Analysis". This is a very short chapter that introduces the various types of ports available and describes the way that information may be passed from the computer to the attached instruments.

2) Discuss the features of the data acquisition board and its connections with computer and instruments. Emphasize the data acquisition board as a vehicle for transferring information.

3) Introduce instruments, both "real" and virtual, and explain how they handle and generate data. At this point the student has the necessary background to understand why languages like LabVIEW are important.

4) Introduce the specifics of the LabVIEW language and have the student build some simple VIs. The VIs in Figs. 1 and 2 are easy to create and can be expanded to use filters or statistics VIs. Also, the student edition gives many good examples of VIs of varying sophistication, and there are numerous exercises with solutions in the teacher's edition.

5) Increasingly sophisticated VIs, designed for real applications, can now be designed. The specifics will depend on the interests of the students and the instructor, the level of the students, and the amount of time allowed for the course. There is a lot of room for creativity because LabVIEW is a powerful program, and little hardware is required.

LEARNING MORE AND TECHNICAL SERVICE

National Instruments has a very good, free tutorial that gives a hands-on introduction to everything needed to write a basic program. The tutorial takes several hours to complete but is well worth the time. There are different versions, depending on the type of computer system you are working on, so be sure to specify which one you want. There is also a tutorial for HiQ.

In addition, several books, a video, and courses on the different aspects of using LabVIEW for interfacing are available. Of these I am familiar only with the video, which is inferior in quality to the tutorials.

Technical service is available through the mail and by E-mail, fax, and telephone. The few times I asked for technical service, I found it to be promptly delivered and accurate.

COMPUTER REQUIREMENTS AND COSTS

LabVIEW is available for various computers including IBM PCs (or clones), Apple Macintosh (from Mac IIs to the present), workstations, Sun Sparcstations, and HP-UX systems. The computer must have a math co-processor, and the IBM-type machines must run Windows. LabVIEW is memory intensive. The Mac version requires 5–8 MB of RAM, depending on what you want to do, and requires 41 MB of disk storage space to install the entire package. Memory requirements for IBM-type machines are similar. The student version does not require the math co-processor but has the same memory requirements as the full-scale version.

The full LabVIEW development system lists for $1995 and the HiQ package costs an additional $695, but there are various discounts available, including an educational discount. The student version of LabVIEW sells for $60 but is only available through Prentice-Hall. Boards vary in both cost and capability. A very nice board that has 12-bit ADC, 62.5 kS/s sampling rate, on-board timers, and digital I/O lines as well as analog-input channels has a list price of $695. You can equip a classroom with two of the boards described and 21 software packages for under $4000. For more information, contact National Instruments, 6504 Bridge Point Parkway, Austin, TX 78730-5030 (Tel.: 512-794-0100).

SUMMARY

For instrumentation-intensive sciences such as biophysics, an object-oriented software approach to computer interfacing makes sense. It is more intuitive and easier to use than standard programming languages, and it is more flexible and more general than most of the company-designed, user friendly computer languages. Teaching this approach to biophysics students is feasible, even within severe budgetary constraints, and will help to make them more high-tech and productive scientists.