



WEB DELIVERY OF INTERACTIVE LABORATORIES: COMPARING THREE AUTHORING TOOLS

By Richard R. Silbar

AS EDUCATORS KNOW, THE MORE A STUDENT INTERACTS WITH A SUBJECT, THE BETTER HE OR SHE WILL LEARN IT.

THIS IS PARTICULARLY TRUE IN TECHNICAL SUBJECTS. ONE WAY TO PROMOTE INTERACTION IS TO HAVE "LABORATORIES" IN

which the student manipulates objects on the computer screen using the keyboard or mouse and then sees those actions' outcome.¹ We at WhistleSoft have built, using Macromedia's Authorware, numerous such laboratories for use in the Accelerators and Beams series of computer-based tutorials. (The "Useful URLs" sidebar lists the Authorware Web site and other pertinent URLs; the "Accelerators and Beams" sidebar describes our tutorial series.)

However, Authorware has two big disadvantages. First, it's expensive; the retail price for Authorware 6.0 is US\$2,699. Second, if you try accessing a Web page that contains an Authorware application, you will probably discover that, to view it, you need to download a rather large, multimegabyte plug-in. Moreover, the Authorware Web player is available only for Internet Explorer and Netscape on Windows and Macintosh computers; Unix and Linux folks are out of luck. Unless you are exceptionally motivated (or some systems administrator has already done this work), you probably won't bother with these annoyances and will instead move on to something else.

One way to create interactive laboratories viewable with Web browsers having Java 1.1 and JavaScript capabil-

ity is to make them as Java applets. For example, Wolfgang Christian and his students at Davidson College have been doing just this.² However, to program in Java involves a considerable investment in learning, and it might be easier for those who don't already know that language to consider other ways to put interactivity on the Web.

For example, almost every modern Web browser already comes with a plug-in for playing files in Macromedia's Flash Shockwave format. (According to Macromedia, the Flash player penetration was 97.4 percent as of September 2001. A Flash player even exists for Unix and Linux computers.) And almost 70 percent of browsers have the Shockwave plug-in for playing shocked Director files. Even if these plug-ins are not already installed, they are still smaller than the Authorware player. So, either Flash or Director might be a better tool if you want to deliver training modules on the Web, especially for a heterogeneous audience. In addition, Director is less than half as expensive as Authorware (Director 8.5 costs US\$1,199), and Flash is considerably cheaper (Flash 5 retails for US\$399).

Therefore, as an experiment, I decided to see how easy it would be to

Useful URLs

Davidson College Java applets:
<http://webphysics.davidson.edu/applets/applets.html>

Macromedia Authorware:
www.macromedia.com/software/authorware

Macromedia Director:
www.macromedia.com/software/director

Macromedia Flash:
www.macromedia.com/software/flash

Physics Academic Software:
www.aip.org/pas

Richard R. Silbar's Web site:
www.whistlesoft.com/~silbar

Vector Addition Laboratory

- **Authorware version:**
www.whistlesoft.com/~silbar/demo/vecadd/index.shtml
- **Director version:**
www.whistlesoft.com/~silbar/demo/director/dirvecs.shtml
- **Flash version:**
www.whistlesoft.com/~silbar/demo/flash/AddVecs.shtml

WhistleSoft tutorials:
www.webassign.net/pasnew/whistlesoft.html

clone our Vector Addition Laboratory (from our Vectors tutorial), using Director or Flash. Another factor in this decision was that I already had these authoring tools on hand and had experience using them.³ (To see a Director piece, "A Rolling Stone Gathers No

Mass,” and a Flash animation of $\mathbf{F} = m\mathbf{a}$, visit the Demo section of my Web site, listed in the sidebar.) Because Authorware uses an iconic flow-line approach while Director and Flash use a stage-and-timeline approach, it is not clear offhand that a close clone is possible. As you’ll see, the Director and Flash versions are similar to, but definitely not the same as, the Authorware version.

The Authorware version

The original Vector Addition Laboratory deals with adding two vectors in the geometric picture. Before entering the laboratory, the student sees the Content page (see Figure 1a), which explains the process. Clicking the Animation button starts an animation with the vectors \mathbf{A} and \mathbf{B} separated, then moves them together, tail-to-head, and finally draws the vector sum, \mathbf{C} . As this happens, the software highlights the relevant text in the paragraph to the right.

The button with the question mark leads to a set of self-test questions. More interesting is the Laboratory button (showing the letter L and an Ehrlen-

meyer flask). Clicking that, the student comes to the Laboratory page itself (see Figure 1b), which shows three randomly placed vector-arrows. Sooner or later, the student will grab the blue triangle on \mathbf{B} with the mouse and drag it so that its tail falls on \mathbf{A} ’s head. Then, he or she will drag \mathbf{C} ’s blue circles to their appropriate places. Clicking on “Click here when done” provides a numerical score (from 0 to 100) of how well the student placed \mathbf{C} . If the student placed \mathbf{C} incorrectly, the laboratory draws a corrected \mathbf{C} (in purple, leaving the student’s black version). It then offers the student the chance to try another case. Students can repeat the exercise as many times as needed at their own pace until they are comfortable with the concept.

Once we learned how to place the draggable handles on the arrows (I thank Harlow Pinson for a useful suggestion in this regard), creating this laboratory was relatively easy. Authorware comes with a rich scripting language (which is stored in software objects called *calc icons*). This language lets you straightforwardly place, erase, and

Accelerators and Beams

This tutorial series comprises five modules: Vectors, Forces, Motion in Electromagnetic Fields, Dipole Magnets, and Quadrupole Magnets. Our publisher, Physics Academic Software, released our first module, Vectors, in late 1997. The latest, Quadrupole Magnets, became available in June 2001. The modules are available on CD-ROM or floppy disks. We originally designed them as stand-alone executable programs for students to use singly or in teams, either in or outside a formal classroom environment. When we started developing them, Web delivery of our training materials had not yet occurred to us.

redraw lines with arrowheads. Here’s how the laboratory uses this language:

1. The Random number generator creates starting points and end-points for \mathbf{A} , \mathbf{B} , and \mathbf{C} , taking care (iteratively) that they lie appropriately on the screen. The software calculates the correct answer for the sum $\mathbf{A} + \mathbf{B}$ and stores it as a hidden vector \mathbf{D} , which it will use to check the student’s answer.

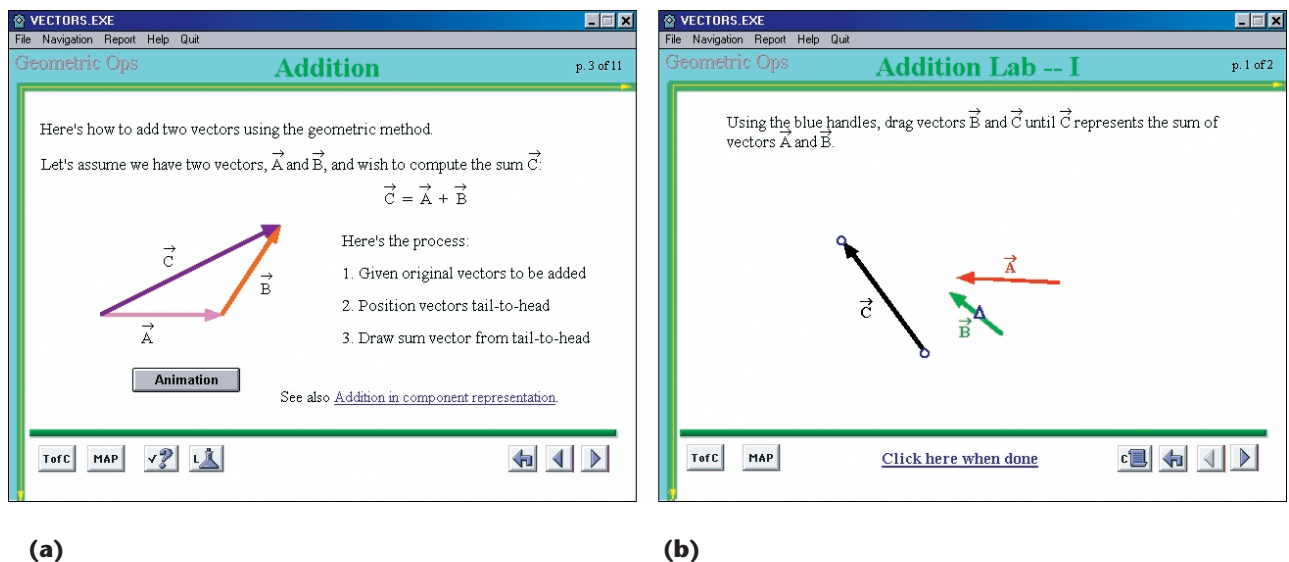


Figure 1. The Authorware version of the Vector Addition Laboratory: (a) the Content page; (b) the Laboratory page.

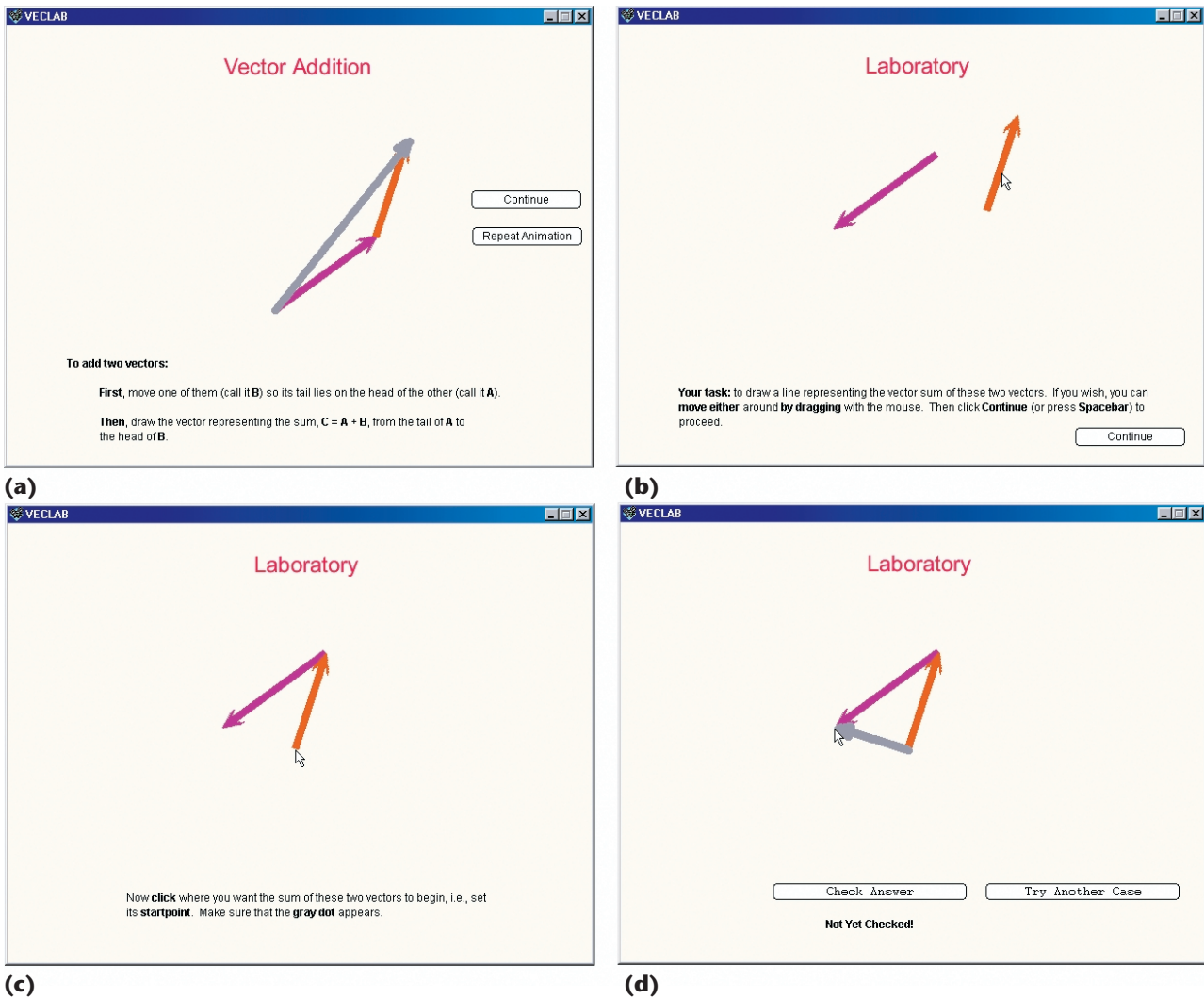


Figure 2. The Director version of the Vector Addition Laboratory: (a) the Content page; (b) the first frame; (c) the second frame; (d) the last frame.

2. The software draws **A**, **B**, and **C** on the screen, using the functions `SetLine(2)` (so that they have arrowheads), `SetFrame` (to set a color), and `Line(pensize, x1, y1, x2, y2)`.
3. The software places the blue triangular handle at **B**'s midpoint and the blue circular handles at **C**'s starting point and endpoint. It also places labels near each vector's midpoint. The handles are separate display icons and are active "hot spots" that respond to mouse clicks and drags.
4. When the student drags a handle, the `EraseIcon` function removes the previous incarnation of that

vector (the calc icon that drew it). It then redraws the vector according to the position of the built-in `cursorX` and `cursorY` variables giving the mouse's position.

5. When the user clicks on "Click here when done," the program flow exits the interaction icon and enters a map icon that animates the correct answer for **C** and calculates the student's score based on the distance between the heads of the student's **B** vector and the correct **C** vector.

Once we built the Authorware application, we compiled it into an executable file (without runtime). We then prepared it for Web delivery using the

Web Packager, a separate piece of software. This involved creating an `aam` file, which tells the Web player what files to download from the server, and a series of `aas` files (segmenting the executable for streaming it to the browser). The total download for the Authorware Web player is 305 Kbytes, but not all of it must be present for the application to begin playing.

The Director version

I decided to make the Director clone first because Director comes with Lingo, an extensive scripting language that I thought might facilitate building user interactions. As I rapidly learned, however, Authorware's scripting language and

```

global vectorsMoveable, whichVec

on mouseDown
    set whichVec = the clickOn
    if vectorsMoveable = FALSE then pass
    moveVector whichVec
end

on mouseUp
    set the moveableSprite of sprite whichVec = FALSE
    updateStage
    pass
end mouseUp

```

Figure 3. Director Lingo scripts for the mouse actions that enable the dragging of each vector object in the cast.

Lingo have very different capabilities. This led to a decidedly different look and feel, even though the Director version's content turned out to be almost the same as the Authorware version. I built this version using Director 6.0. The Shockwave file is 47 Kbytes, quite a bit smaller than the Authorware download. In the spirit of a scientific experiment, I did not try to make the graphics for this version pretty.

This version starts with a frame (see Figure 2a) that is roughly equivalent to the Authorware Content page. This page opens with an animation in which **B**, the brown vector, moves into position from the left. Then the application draws a gray arrow representing the vector sum **C** from the tail of **A** (purple) to the head of **B**. Users can repeat the animation as often as desired, comparing the motions with the instructional text below.

When the student is ready, he or she clicks the Continue button and proceeds to the next page, which is the laboratory itself. It unfolds in a succession of frames; Figure 2b shows the first. The screen snapshot shows the student about to drag the brown vector into position, with its head at the purple one's tail. When the student has positioned the vector, clicking on the Continue button brings up the next instruction (see Figure 2c). Here the student is about to click where he or she wants the sum vector to begin. When this happens, a gray dot appears at that point, and the next frame, for setting the sum vector's endpoint, appears. Figure 2d shows what the screen looks like just after the student has clicked at the desired end location and the software has drawn the resulting gray arrow.

From here the student can check the answer or try a different random case. Unlike the Authorware version, this version does not give a numerical grade.

Lingo's limitations necessitate a different evaluation of the goodness of fit. An example of a grade is a text response such as "Good!" or "Perhaps in the wrong direction? Try again."

How did I program the Director version? Drawing a line is difficult using Lingo, which does not have built-in functions like Authorware's `Line(...)` and `SetLine(...)`. So, I placed the initial vectors on the stage by choosing randomly from 80 different arrows (four sizes, each in 20 different orientations) in the Cast (a library of displayed objects, scripts, and so on). Using the built-in `AutoDistort` tool, I generated the different orientations for each size from an initial horizontal vector.

Dragging the individual vectors was relatively easy because of Lingo's `mouseDown` property. Each of the 80 vectors in the cast has the script shown in Figure 3. The variables `vectorsMoveable` and `whichVec` are global because they are useful in other scripts for this application.

A big difference between the Authorware and Director versions is in how they draw the sum vector **C**. The Director version places a series of gray dots at calculated steps between the starting point and endpoint the student specifies. To draw the arrowhead, the application places dots, using trigonometric calculations to find their locations so that the arrowhead points in

the right direction. Frankly, the gray arrow for **C** is a bit sloppy, and it was awkward to program.

Finally, the Check Answer button has a lengthy script attached that calculates a figure of merit based on the differences between the magnitude and angle of **C** given by the student and those determined from the initially chosen vectors **A** and **B**. This also involves numerous (awkward) Lingo lines such as

```

set locVecX =
    the locH of
    sprite whichVec

```

where `locVecX` is a local variable and `locH` is a Lingo property.

I've investigated reauthoring the laboratory in Director 8.5, and I do not believe that this would improve its interactivity much. The Lingo enhancements since Director 6.0 are concerned mostly with other features and capabilities than the ones needed here.

On to Flash

The Flash version also begins with an animation (Content page) showing the student how to add two vectors. This was easy to do using symbols for the three vectors and using *motion tweens*, one of Flash's strengths. (*Tweening*, a concept perhaps invented by Disney, involves creating cel frames that interpolate between two key frames, which

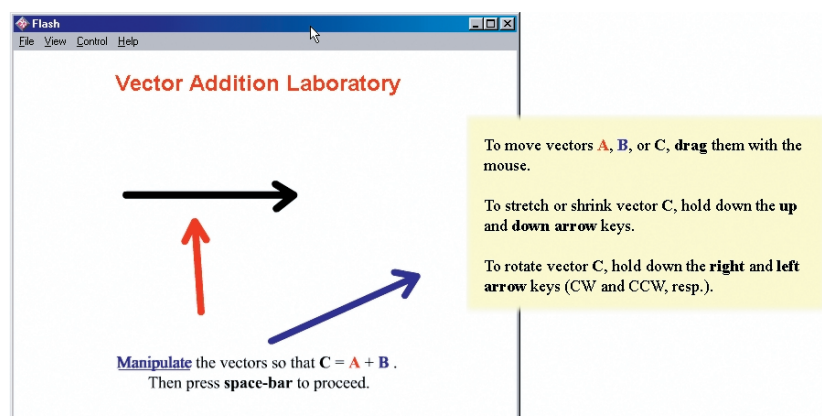


Figure 4. The Flash version of the Vector Addition Laboratory. The yellow text box on the right pops up when the student clicks on the blue, underlined “Manipulate.”

here give the starting and final positions of the object that moves.) Because this Content page is not much different from the Authorware and Director Content pages, it is not displayed here.

After entering the Lab, the student sees three randomly placed vectors, **A** (red), **B** (blue), and **C** (black) (see Figure 4). Clicking on the blue “Manipulate” link brings up a yellow pop-up window (shown on the right) explaining how to manipulate these vectors. As in the Director version, students can use the mouse to drag the arrows around the stage. In addition, students can use the arrow buttons on the keyboard to resize and rotate **C**.

When the student is satisfied with **C**’s position, pressing the space bar leads to a new frame that lets the student check his or her answer or try another case. Pressing the Check Answer button gives an evaluation of “Good!” or “You can do better. Please try again.”

My first attempt to build this laboratory used Flash 4, which has a much less rich scripting language than Flash 5. As a result, that version had a klunky user interface with a bunch of buttons for incrementing positions, angles, and sizes. The simplifications in the Flash 5 version, shown in Figure 4, are due to enhancements of the action scripting, which now looks much like JavaScript. In particular, adding the `onClipEvent` action allowed a continuous flow of rotations and rescalings for **C**. Also, I im-

plemented the dragging of the separate vectors by incorporating a button in the movie clip for each vector. This lets the movie clip accommodate an `on(mouseEvent)` to set the separate `startDrag` and `stopDrag` conditions—for example,

```
on (press)
{startDrag("/vectorB", true,
100, 75, 450, 325);}
on (release)
{stopDrag();}
```


(The arguments in the `startDrag` function confine the dragging to a restricted region of the screen.) Before I learned to incorporate the button in the movie clips, students had to hold down the “a,” “b,” or “c” keys to distinguish which vector was draggable—also an awkward maneuver.

Checking the student’s answer for **C** again involves a script with much trigonometry, such as

```
C_x = Cmag*Math.cos(Crot *
Math.PI/180.),
```

where `Cmag` and `Crot` are variables calculated earlier in the script. One peculiarity of Flash’s geometrical conventions is that positive angles (in radians) are clockwise rather than counterclockwise.

The Flash 5 version involves a 14-Kbyte download, much smaller than either the Director or Authorware version.

Selecting an authoring tool comes down to a matter of taste, cost, and how hard you want to work. It was easiest to create the Vector Addition Laboratory (and others like it) with Authorware, whose scripting functionality provides much flexibility. Director and Flash are almost as capable for creating this kind of interactive laboratory, but the user interfaces are not as intuitive or clean. However, bearing in mind Authorware’s plug-in problem for Web delivery, you might choose to accept the compromises and use Director or Flash. 

Acknowledgments

J. Patrick McGee, Robert A. Williams, and William C. Mead collaborated with me to build the original Vector Addition Laboratory. The encouragement to make Director and Flash clones came from Dixon Wolf, who also made several valuable suggestions about how to proceed.

References

1. E.F. Redish, J.M. Saul, and R.N. Steinberg, “On the Effectiveness of Active-Engagement Microcomputer-Based Laboratories,” *Am. J. Physics*, vol. 65, no. 1, 1997, pp. 45–54.
2. W. Christian and A. Titus, “Developing Web-Based Curricula Using Java Physlets,” *Computers in Physics*, vol. 12, no. 3, July/Aug. 1998, pp. 227–232.
3. R.R. Silbar, “Animating Equations on the Web,” *Computing in Science & Eng.*, vol. 2, no. 4, July/Aug. 2000, pp. 91–95.

Richard R. Silbar worked as a theoretical nuclear physicist at the Los Alamos National Laboratory before forming WhistleSoft, a software development company specializing in multimedia products for science and engineering. He earned his BS, MS, and PhD in physics from the University of Michigan. Contact him at WhistleSoft, 168 Dos Brazos, Los Alamos, NM 87544; silbar@whistlesoft.com.