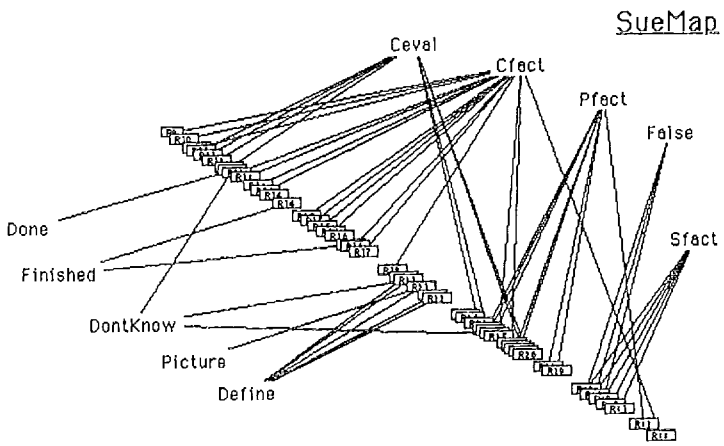

Figure 7. Coding of Sue's retelling of a story

References

Fielding, N. G., & Lee, R. M. (Ed.)., 1991. *Using computers in qualitative research.* London: Sage Publications.

Goldman-Segall, R., 1990. "Learning constellations: A multimedia research environment for exploring children's theory-making." In I. Harel (Ed.), *Constructionist learning* , pp. 295-326. Cambridge, MA: MIT Media Laboratory.

Horney, M. A.,1992. "Supporting multiple perspectives: Case studies of using hypertext in qualitative research." In M. Brown (Ed.), Proceedings of the Qualitative Research in Education Conference, 1992. Athens, Georgia: University of Georgia.

LeCompte, M. D., Millroy, W. L., & Preissle, J. (Eds.)., 1992. *The handbook of qualitative research in education.* San Diego: Academic Press, Inc.

Research–Guided Design of Multimedia Research Tools

Robert J. Beichner
North Carolina State University
Physics Department
Box 8202
Raleigh, NC 27695
(919) 515-7226
Email: Beichner@NCSU.edu

(This work is partially funded by the National Science Foundation, MDR-9154127. Additional support came from RasterOps Corporation, Sony Corporation of America, and Apple Computer.)

Since this issue of the newsletter features both multimedia designers and research-oriented users, I thought it would be interesting to merge the two viewpoints and describe my approach to the design of some of the multimedia software I use in my research. The main point of this article is to describe how the findings of prior research can be used to guide the design of software which itself is to be used for additional research. The beginning of the discussion will center around the development of a C program called "VideoGraph." The name furnishes a hint as to what the program does—it provides introductory physics students with a tool for graphically analyzing video sequences. A second package was designed to provide a multimedia editing environment for use by middle school students. Although these two projects sound very different, the software for both shares the same purpose—it gives students multimedia tools to help them learn science topics while minimizing the distraction of the software itself. The research investigated the educational impact of the software.

VideoGraph

The main task of the VideoGraph package is to replay a video of one or more moving objects while synchronously generating a graph of position or velocity. Many studies have shown that students have a great deal of difficulty making the cognitive leap from the concrete world of motion events to their abstract mathematical—in this case, graphical—representations (Barclay, 1986; McDermott, Rosenquist, & Zee, 1987; Mokros & Tinker, 1987; van Zee & McDermott, 1987; Brasell & Rowe, 1989; Beichner, 1993). Students tend to view graphs as a sort of photograph of the scene. For example, when asked to sketch a velocity versus time graph for an object rolling down a hill, across a flat area and then back up a hill, the resulting drawing will often reproduce the hills and valley traversed by the object. When shown the actual kinematics graphs, students are not able to interpret their meaning.

Besides the problems students have in understanding kinematics graphs, they also have some basic difficulties with other aspects of Newtonian mechanics (Trowbridge & McDermott, 1980; Trowbridge & McDermott, 1981; Halloun & Hestenes, 1985; Hestenes, Wells, & Swackhamer, 1992). For example, students believe (even after traditional instruction) that there must be a force on an object if it is moving. The classic test for this is to toss a coin into the air and ask students to describe any force which might be acting on it during its flight. When interviewed, the common response is to describe a "force from the throw" that decreases as the coin rises. The peak of the projectile's motion takes on an inappropriate level of importance to these students when in fact the acceleration of the coin remains constant throughout the throw. (A graph of the velocity is a straight, downward sloping line which crosses the time axis at the instant the coin reaches the top of its arc.)

Rather than turn this article into a physics lesson, let us briefly turn our attention to some of the psychology behind the user interface of VideoGraph. Keep in mind that the overall plan for the software is to address the problems research has uncovered about students trying to make the mental links between concrete events and the related graphs. This is directly reflected by the decision to have two main windows, one for the event and the other for a graph.

The video window utilizes QuickTime™ to present a movie of the motion. It is placed directly beside the graph window. Because short term memory is limited in what it can hold and how long its contents can be maintained (Hulse, Egeth, & Deese, 1980) , I decided to keep the motion event directly linked to an adjacent graphical representation. Students can easily go back and forth between the two "worlds." These two displays are synchronized so

that as the movie plays, a circle simultaneously moves from point to point on the graph. By clicking on the single-frame advance controls of the movie, the graphed circle hops to the next point. Alternately, by clicking on a graphed data point, the movie instantly jumps to the frame associated with that point. Promoting this easy transition between motion and graph should help students connect the two in their minds (Shuell, 1986) without strengthening the "graph as picture" error. (In fact, one of the exercises I have students do is adjust the graph so that it displays vertical position versus horizontal position. This is the one case where the graph really is like a photograph of the event. The trail of the object exactly matches the graphed line. I ask students to compare this graph to standard graphs with time on the horizontal axis.)

Another important part of the design consideration was the ease of use of the software. I wanted students to be able to concentrate on the physics being examined, not the operation of the program, so I tried to make the controls as direct-acting as possible (Shneiderman, 1992). For example, to adjust what is being graphed, students click the mouse on either the vertical or horizontal graph label for a pop-up menu of possibilities. Clicking on an axis number produces a small box at that spot for entering a new upper or lower limit for that axis or new value for grid spacing. In situations where this direct interaction with what is to be affected isn't possible, I tried to use controls that were already familiar to the students. For example, the standard QuickTime window provides a triangular play button. VideoGrab, the companion package for capturing video data, has similar controls for playback, recording, and advancing a videodisc player or Visca-capable VCR. A single slide control switches operation from one piece of video equipment to the other. This capture control window was originally part of VideoGraph but was removed to a separate program since many students would not be capturing video, the necessary hardware being quite expensive. (Apparently this separation of tasks is helpful even when students perform both the capture and analysis of data. In field tests, students quickly used VideoGrab to collect their video and then went on to spend most of their time analyzing it with VideoGraph. Because VideoGrab can be set up to automatically grab any number of images—skipping as many frames between each capture as desired—students are able to complete this step in just a minute or two.)

Another study (Lea, 1993) has shown that students are able to accurately interpret strobe photos of motion events. This ability was exploited by VideoGraph by allowing students to display all location markers on each frame. Of course, this can be turned off so that small objects can be seen as they move across the screen. Humans have evolved a sensitive motion detection and tracking facility (Wallach, 1959). Our pattern recognition skills are also extremely powerful. VideoGraph was designed to take advantage of both of those capabilities in order to improve instruction.

Besides reproducing motion events that are easily seen in everyday life, it is also possible to use VideoGraph to provide insight into situations not generally accessible to casual observers. For example, it is possible to lock an object in place while the rest of the scene moves around it. In other words, students can switch to the reference frame of a moving object. (If an object which jiggles around in the video but is actually stationary is "locked down," the software corrects for camera motion and panning and still allows students to follow the action.) VideoGraph can also determine and track the center of mass of a group of objects, allowing students to watch as the center of mass of a high jumper always stays under the bar, for example.

These visual enhancements are supplemented by additional graph calculation algorithms. Students can quickly and easily determine slopes of different parts of the graphed lines as well as examine areas under the curves. These graph characteristics have important kinematic interpretations which are often not made by students because of difficult or lengthy calculations.

One final aspect of this package which might be worth mentioning is that students actually enjoy using it. By giving them access to commercial videodisc images, they can utilize unusual motion sequences—a jet takeoff, overhead view of a baseball player swinging at a pitch, automobile collisions, etc. Motivation and ego-involvement are increased by also being able to incorporate student-captured images of familiar situations.

VideoGraph is now in use in half a dozen college and high school classrooms as part of a research project aimed at improving the teaching of kinematic graph interpretation. Standardized graphing test results and individual interviews will be compared to the performance of students using a more traditional curriculum. The early results are promising.

Multimedia authoring

The second piece of software to be described here is called MAST, an acronym for Multimedia Authoring for Students and Teachers. Its purpose is to provide a powerful editing environment for combining text, graphics, video, and audio. In keeping with the overall goal of supplying tools which don't obscure the topic of instruction, it was designed to be as easy to operate as possible. Outwardly, its purpose was to help middle school students create information screens for a touch-sensitive kiosk at a local zoo. Of course, the hidden reason for its use was the hope that students would learn a substantial amount of science while editing the multimedia materials. Students were videotaped and interviewed as a source of corrective feedback on the user-interface and to document the learning process.

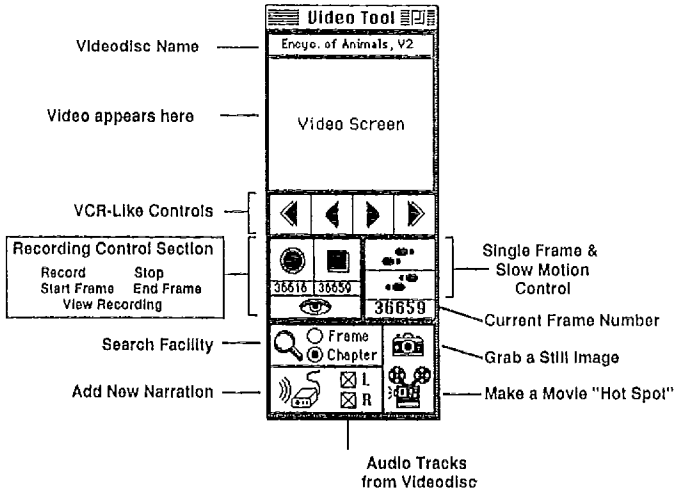

Figure 1. The video tool for MAST provides a quick way to control a videodisc player and capture still frames or movie sequences.

There were several aspects of the setting of the MAST project that make it unusual. First of all, the school was located on the grounds of a large zoo. Zoo staff and school personnel worked cooperatively, to the benefit of all. This tremendous resource was supplemented with a very large array of technology, including videodiscs, CD-ROMs, scanners, electronic cameras, MIDI devices and even robots. Students beginning the project were highly computer literate. (In fact, they were so accustomed to using productivity software that they asked that a slider that had been designed to simplify the sizing of text be replaced with a more traditional font menu.)

The customized editing environment that was developed for student use was actually a hybrid. Most of the functionality was created with HyperCard, but when color painting or text was

required, a "behind the scenes" switch to a stand-alone SuperCard application (with the same menus as the HyperCard main editor) was made. Students were not even aware that they were working in two different programs. An audio tool was provided by HyperCard's built-in recording facility. (Early efforts at an editor done complete in SuperCard were stymied by the difficulty of creating a reliable audio recorder.)

This was complemented by a video tool, shown in figure one. Both these tools generated "hot spots" on the screen. These spots were actually text fields with associated scripts to make them touch sensitive (essentially responding to a mouse click). Students would add appropriate text like, "Touch to hear an elephant!" or "Touch here to watch gorillas playing." A linking tool was also provided to allow quick hypertextual connections to other screens. Besides making plain text spots which were not responsive to touch, students also were able to create hotspots which would print out materials for zoo visitors. These materials typically would contain a map and a series of student-generated questions that could be answered while viewing the exhibits. The students requested the inclusion of a smell tool that would recreate the "aura" surrounding many animal displays, but this was not implemented!

(An interesting aside came up because the students persisted in saying they were "programming" when they were editing. In a way they were. Clicking on the movie-making button on the video tool resulted in a customized HyperTalk script that was over a page long being stored with a hot spot—the video tool acted like a code-generator. Perhaps we need to expand our ideas of what programming really is. For example, when we use graphics software to draw a rectangle, aren't we actually programming in both a screen-display language like QuickDraw and a printer language like PostScript? We may not know the syntax details, but we are generating a series of steps to be carried out by the computer.)

Regardless of the purpose of the hot spot, these special text fields could be moved around the screen by holding down the shift key while dragging them with the mouse. (The touch screen was only used on the actual kiosk, not the editing workstations.) Pressing the option key while clicking on a hot spot allowed the student editors to resize and reshape it. Holding down the command key while clicking would delete—after a confirming dialog box—the hot spot and any resources (sounds, PICTs, etc.) associated with it. These three keys were used consistently throughout the editor, not just for hot spots. For example, they worked in similar ways on color graphics whether they were still frames captured from video, images recorded with a digital camera, pictures collected by a flatbed scanner or paintings made by the students.

The purpose of the careful design was to provide a tool that would let students work on what they saw as an important task—a highly motivating situation which should lead to increased learning (Cohen & Riel, 1989; Verschaffel, Hoedemaeders, Schrooten, & Indemans, 1988). If the software was transparent enough, students could concentrate on what they were doing and not how they were doing it. We wanted to watch how they worked together, what resources they took advantage of, and what qualities they looked for in multimedia information before they decided to incorporate it into their screens. Working together, as promoted by the software, should improve their learning (Johnson, Maruyama, Johnson, Nelson, & Skon, 1981). We also hoped that their review of resources available to them would help students build connections between concepts and generally learn the content (Duell, 1986). Other researchers have seen the importance of students constructing their own knowledge (Piaget, 1954; Magoon, 1977).

There was also reason to believe that having the children actually create something useful would have a positive impact on learning (Florio, 1979; Papert, 1990). Our transcriptions of the videotaped editing sessions and interviews showed that this was indeed the case. Students were extremely interested in the multimedia screens they were creating and the audience they were creating them for. The students were surrounded by technology, but it was the task that provided the extra motivation for this project.

Conclusions

This article described the design of two very different multimedia software packages that were used for educational research. This research-informed design methodology resulted in packages which worked very differently, but accomplished the same goal—to non-obtrusively expand students' capabilities in order to improve learning.

References

Barclay, W. L., 1986. "Graphing Misconceptions and Possible Remedies using Microcomputer-Based Labs." Paper presented at the National Educational Computing Conference, San Diego, California.

Beichner, R., 1993. "Testing student interpretation of kinematics graphs." Manuscript submitted for publication.

Brasell, H., and Rowe, M. B., 1989. "Difficulties in Constructing and Interpreting Graphs." Paper presented at the annual meeting of the American Educational Research Association, San Francisco, California.

Cohen, M., and Riel, M., 1989. "Effect of distant audiences on students' writing." *American Educational Research Journal*, 26, pp. 143-159.

Duell, O., 1986. "Metacognitive skills." In G. Phye & T. Andre (Ed.), *Cognitive Classroom Learning: Understanding, Thinking, and Problem Solving*, pp. 205-242. Orlando, FL: Academic Press.

Florio, S., 1979. "The problem of dead letters: Social perspectives on the teaching of writing." *Elementary School Journal*, 80, pp. 1-7.

Halloun, I. A., and Hestenes, D., 1985. "Common Sense Concepts About Motion." *Am. J. Phys.*, 53, pp. 1056-1065.

Hestenes, D., Wells, M., and Swackhamer, G., 1992. "Force concept inventory." *Physics Teacher*, 30, pp. 141-158.

Hulse, S., Egeth, H., and Deese, J., 1980. *Psychology of Learning*. New York: McGraw-Hill.

Johnson, D. W., Maruyama, G., Johnson, R. T., Nelson, D., and Skon, L., 1981. "Effects of cooperative, competitive, and individualistic goal structures on achievement: A meta-analysis." *Psychological Bulletin*, 89, pp. 47-62.

Lea, S., 1993. "Assessing degrees of student understanding of acceleration." Paper presented at summer meeting of the American Association of Physics Teachers, Boise, ID.

Magoon, A., 1977. "Constructivist approaches in educational research." *Review of Educational Research*, 47, pp. 651-693.

McDermott, L. C., Rosenquist, M. L., and van Zee, E. H.,1987. Student Difficulties in Connecting Graphs and Physics: Examples from Kinematics. *American Journal of Physics*, 55, pp. 503-513.

Mokros, J. R., and Tinker, R. F.,1987. "The Impact of Microcomputer-Based Labs on Children's Ability to Interpret Graphs." *Journal of Research in Science Teaching*, 24, pp. 369-383.

Papert, S., 1990. Introduction. In I. Harel (Ed.), *Constructionist Learning* , pp. 1-8. Cambridge, MA: MIT Media Laboratory.

Piaget, J., 1954. *The Construction of Reality in the Child.* New York: Basic Books.

Shneiderman, B., 1992. *Designing the User Interface: Strategies for Effective Human-Computer Interaction.* Reading, MA: Addison-Wesley.

Shuell, T. J., 1986. "Cognitive Conceptions of Learning." *Review of Educational Research*, 56, pp. 411-436.

Trowbridge, D. E., & McDermott, L. C., 1980. "Investigation of student understanding of the concept of velocity in one dimension." *American Journal of Physics,* 48, pp. 1020-1028.

Trowbridge, D. E., & McDermott, L. C., 1981. "Investigation of student understanding of the concept of acceleration in one dimension." *American Journal of Physics,* 49, pp. 242-253.

van Zee, E. H., and McDermott, L. C., 1987. "Investigation of Student Difficulties with Graphical Representations in Physics." Paper presented at Seattle, Washington.

Wallach, H., 1959. "The Perception of Motion." *Scientific American,* 201(1), pp. 56-60.

Analyzing User Interactions with Hypermedia Systems

Wayne A. Nelson
Dept. of Educational Leadership
Southern Illinois University at Edwardsville
Edwardsville, IL 62026
(618) 692-3286
Email: wnelson@siuevm.bitnet

The unprecedented freedom for users to control the scope and sequence of their interactions with hypermedia systems presents many challenges to those who design and study these systems in educational settings. Early efforts to develop hypermedia systems revealed that the node-link structure of such systems is both advantageous and problematic (Conklin, 1987). When users have the freedom to follow any of a multitude of link permutations, disorientation often results. Further, without appropriate training, novice users do not possess the strategies necessary for effective "browsing" of large hypermedia documents (Duffy & Knuth, 1991). It has also been noted that the purpose for using the system or the task that the users are engaged in can influence patterns of interaction with hypermedia systems (Nelson, 1991). Many designers, therefore, advocate that features such as visual maps, database search facilities and guided tours be included in hypermedia systems to alleviate some of these problems (e.g. Hammond, 1989; Laurel, 1990, 1991).

With the emergence of hypermedia systems as a major architecture for educational and other information-oriented software comes the related problem of how to document and analyze user interactions with such systems for the purposes of research and evaluation. There are a variety of interface design strategies that impact on how a system performs and should be evaluated. Many hypermedia systems to date have employed a "browsing" interface, but alternative approaches are also emerging (Nelson & Palumbo, 1992). Regardless of the type of interface, many questions can be generated when studying the interactions of users with hypermedia systems. For example, how many users chose to follow a particular link, and why was one link chosen over another? How does the choice of one link affect choices of subsequent links? When are graphic images, animations and video segments accessed? What user tasks are appropriate for guiding interaction with the system? What kinds of strategies do users develop while working with hypermedia systems? These and many other questions need answers when designing, developing and evaluating hypermedia applications for education and other settings, and provide the focus for this short article.

A wealth of user interaction data can be easily collected within many hypermedia development environments in order to study aspects of the interface, including the nature of user navigation patterns, the time spent at each node and the use of help and orienting facilities. The data can represent the paths a user follows through the system, and the choices made at each node in the system. The problem is that because of the nature of this data, traditional methods of analysis such as surveys or pretest-posttest designs, are not particularly effective for determining usability or comparing alternative interface designs. Researchers have had to develop new techniques for analyzing patterns of user interaction in order to evaluate the design and effectiveness of hypermedia systems (Misanchuk & Schwier, 1992). There is a need to categorize and compare groups of users in order to compare the effectiveness of alternative system features, as well as describing characteristics of interaction by individual users within the same system.

Characterizing user interactions using path algebras

Characterizing the interactions of individual users, or comparing groups of users, can be accomplished using several methods derived from mathematical set and graph theories (Backhouse & Carré, 1975; Carré, 1971). In the first method, path algebras are used to describe and compare the routes users take through hypermedia systems (Alty, 1984). User interactions are characterized in terms of various types of paths through a two-dimensional space (Canter, Rivers & Storrs, 1985). The moves from node to node of each individual can be recorded in data files while the user interacts with the system. For example, a simple Hypercard script such as that shown in Figure 1 will produce a data file indicating the sequence of nodes visited by the user. Using similar programming techniques, it would also be possible to "trap" each user action at individual nodes, such as clicks on buttons, menu selections, or viewing graphic images, animations, or audio and video segments, and save these actions in data files as well.

Scripts for collecting and saving user data	
data file	**Hypothetical**
on openCard	1
global pathVariable	12
put the number of this card & return after pathVariable	13
end openCard	7
	8
on closeStack	6
global pathVariable	4
open file "User data"	8
write pathVariable to file "Userdata"	9
close file "User data"	14
end closeStack	9
	11
	2
	.
	.
	etc.

Figure 1. HyperTalk scripts for collecting and saving user path data, and the resulting data file