11-1-2007

# Designing an Effective Web-Based Coding Environment for Novice Learners

Heejung An

Follow this and additional works at: http://nsuworks.nova.edu/innovate

Part of the Education Commons

This Article has supplementary content. View the full record on NSUWorks here:
http://nsuworks.nova.edu/innovate/vol4/iss1/3

# Designing an Effective Web-Based Coding Environment for Novice Learners

All exhibits, tables and figures that have remained available have been included as additional content with their respective articles to be downloaded separately. Click here to return to the article page on NSUWorks and view the supplemental files.

Unfortunately, not all the supplemental files have survived until 2015 and some will be missing from the article pages. If you are an author in Innovate and would like to have your supplemental content included, please email the NSUWorks repository administrator at nsuworks@nova.edu.

# Designing an Effective Web-Based Coding Environment for Novice Learners
*by Heejung An*

At many technical community colleges, students are seeking to master new skills, such as a familiarity with Web design and scripting, in an effort to obtain employment upon graduation. Yet unlike their counterparts in traditional four-year college programs, these students move through the curriculum at a rapid pace and are rarely provided with an adequate means to practice the hands-on aspects of their trade in a structured classroom environment. However, it is precisely this emphasis on practice that can enable beginners to gain more expertise.

While teaching HyperText Markup Language (HTML) and multimedia courses at Technical Career Institutes in New York City, I observed how quickly students moved through the program, and I began to think about how I could provide supplementary learning tools in addition to face-to-face instruction. In doing so, I wanted to provide students with the opportunity to practice HTML scripting at their own pace and subsequently gain a better understanding of the structure and logic of this code for their future work. After a few semesters of experimentation, I developed an interactive online coding environment, called the Virtual Lab, that enabled learners to write and generate their own code and receive instant feedback in order to improve their coding skills. HTML was selected as the topic of choice primarily because it was considered to be the introductory boot camp course for each first-year class. Overall, the Virtual Lab was intended to become an online personal tutor as well as a supplementary tool for face-to-face instruction.

In designing the Virtual Lab, I created a mechanism to help my students think about the purpose of each action and correct conceptual problems that they often encountered while trying to improve their coding skills. For this undertaking, I also wanted to prevent them from merely typing the code verbatim and then immediately seeing what was produced, which would only develop their factual knowledge and motor skills (Carroll 1984). The aim of this article is to describe the theoretical principles that guided my design decisions for this endeavor and to discuss the results of a preliminary formative evaluation of the pilot program.

## Program Design

*Theoretical Principles Informing the Instructional Design of the Virtual Lab*

The design of the Virtual Lab was guided by the belief that learning from worked examples combined with problem-solving (learning-by-doing) activities would be the most effective approach; indeed, scholarly research on these pedagogical methods suggests that they would work best in conjunction with one another during the initial stages of cognitive skill acquisition (Exhibit 1). Additionally, the design was guided by prior research on the value of immediate feedback in computer-assisted learning environments, which suggested that such feedback can substantially enhance learning and improve student performance (Exhibit 2). While such findings do not apply to all academic disciplines or subjects, they were particularly suited for the distinctive proficiencies that I sought to instill in my students.

In creating the example and problem elements, I also considered how the examples themselves could be sequenced and arranged (inter-example features) to ensure their optimal pedagogical effectiveness (Atkinson et al. 2000). Prior studies provide some helpful guidance in this area as well. Research conducted by Reed and Bolstad (1991) indicates that one example may be insufficient for helping a student induce a usable idea. These authors suggest that the generation of a second example illustrating the idea—especially one that is more complex than the first—creates significant benefits for the transfer of knowledge. In another study, Trafton and Reiser (1993) tested the pairing of examples and practice problems created for a LISP

programming curriculum. They found that a lesson in which each worked example was paired with a practice problem in an alternating sequential pattern throughout the practice sessions produced better learning outcomes than a lesson in which a series of examples was followed by a series of practice problems. Such findings established a further basis for my instructional design of the Virtual Lab.

*Design of the Virtual Lab*

In line with the literature relevant to these instructional principles, the Virtual Lab pilot program consisted of one module (HTML tables) and was developed using JavaScript, HTML, Cascading Style Sheets (CSS), and Active Server Pages (ASP). The program was housed on a Windows 2000 server.

The architecture of the learning environment provides a pathway along which the student advances through successive stages of the lesson (Figure 1). Before entering the Virtual Lab, students are first provided with a general introductory lesson on HTML tables; this lesson provides a basic foundation for their subsequent activities (Figure 2). Upon entering the Virtual Lab, students then work their way through four example-problem pairs, which further illustrate different elements of HTML coding for tables and require students to perform their own coding operations.

As one element of the design, worked examples are provided so that participants can use them as models for solving similar problems. In designing these examples, I determined that novice learners had not yet developed a full sense of the code structure and the relationship between variations in the source code and their associated outcomes (Petre 1995). Thus, in developing step-by-step procedures in the examples, I incorporate typographical enhancements—often referred to in the literature as "pretty printing" (Ledgard 1975)—as an aid to students. In particular, specialized typefaces and indentations help students visualize the code structure, and colored fonts help students develop a better understanding of how each line of code relates to its associated output. In addition, each step contains short descriptive prompts that associate specific modifications in the code to specific features of the output (Figure 3).

In turn, each example in the Virtual Lab is followed by a related problem activity in which students apply what they have learned in their own authoring of HTML code. As students solve these problems, they are prompted to write a coding sequence for a particular table design in the left-hand portion of the interface; they can then click on "submit my code" to see the result of their work and get immediate feedback on their work ( Figure 4). For the immediate feedback component, I embedded a variety of hints and cues to assist learners in determining the correct responses to the problem (Nielsen 1990). In this way, the feedback element ensures that everyone receives individualized assistance and that students who made any errors in their code can repeat the problem again (Figure 5). However, taking into account the potentially negative effect of continual mistakes (Clariana 2000), I also determined that multiple-try feedback should cease at the point when the learner made several failed attempts to solve the problem. I therefore designed the Virtual Lab problems to allow five chances to submit code with immediate directive feedback on errors; after the fifth incorrect submission, the learner is then provided with the correct code in the program error response box and encouraged to move to the next example. Through this aspect of the design, the problems can boost and scaffold the coding activities of the student without creating a high level of frustration.

Finally, the four example-problem pairs are sequenced so that their complexity increases from the first to the fourth pair. The first example-problem pair pertains to creating a table without any spanning cells, the second pair deals with tables that include a cell spanning across the columns, the third pair addresses tables where the cell spans down the table rows, and the fourth pair focuses on tables where the cells span both across the columns and down the rows (Figure 6). My decision to increase the complexity was based on the study conducted by Reed and Bolstad (1991), which had also suggested that example and problem sets be arranged in a sequence of increasing difficulty. Just as the examples serve as a scaffold for their corresponding problems, so too does each example-problem pair establish a level of proficiency that can be further expanded in the following pair.

## Preliminary Formative Evaluation

As this pilot program was being developed, I conducted a preliminary formative evaluation study on the ongoing design process. The goal of this evaluation was to determine the overall efficacy of the Virtual Lab as a learning tool with the purpose of obtaining feedback regarding (a) the perceived usefulness of the program and its particular components and (b) the usability of the program.

*Participants and Study Procedure*
*The fifteen first-year students who participated in this study were enrolled in the Digital Media Arts Technology (DMA) program at Technical Career Institutes in New York City. All of the students volunteered to participate. None of the participants had taken any previous programming or scripting courses. They were familiar with very basic HTML concepts introduced during their current course but not with the creation of tables.*
*The students were tested individually on PC desktop computers with Web browsers utilizing Microsoft Internet Explorer 6.0. All monitors had 18-inch screens. Each student first read the introductory Web-based lesson regarding the creation of tables and then used the Virtual Lab. When students completed their tasks in the Virtual Lab, they were asked to raise their hands. A survey was then administered in pen-and-paper form. There was no time limit for exploration of the program or completion of the survey.*

*Data Collection*
*Methods of inquiry included the survey as well as my own observations. Eleven items were developed for the survey; ten of these items were closed-ended, using a 5-point Likert scale. For the survey, two themes were organized in the following sequence: (a) usability (overall reaction to the program, screen design and navigation) and (b) perceived usefulness of the overall program and the program's particular components. One additional item was open-ended, requesting users to provide comments pertaining to the program (* Exhibit 3*).*
In addition to the survey, each student was observed during program exploration. Observation was used as a supportive and supplementary technique to collect data that could complement and set in perspective data obtained by the survey (Robson 1993). Various aspects of student behavior were observed. In the field notes, any signs providing evidence of the following were noted: (a) students having difficulty or uneasiness in navigating through the program, (b) students attempting to obtain help when learning about the content within the program, and (c) students providing spontaneous verbal opinions about the program. This verbal commentary was unstructured and helped to capture student reactions about the program.

*Data Results*

After the survey was completed, the responses to the closed-ended items were summarized and tabulated ( Figure 7). Responses to the open-ended question were also reviewed and cross-referenced with the quantitative results for the closed-ended questions. In the commentary below, specific topics from the survey questions are highlighted in terms of their numerical ratings on the five-point Likert scale along with additional segments of qualitative feedback from the students.

*Perceived Usefulness*: Overall, participant feedback pertaining to the program's perceived usefulness indicated that it was a valuable tool. For example, students felt that the program was helpful (4.47) and that the three components of the Virtual Lab—worked examples (4.13), hands-on coding problems (4.27), and immediate feedback (4.47)—were very helpful to use during coding practice. Student responses to the open-ended questions further confirmed these results (Exhibit 4).

Judging from responses to survey question 7 (4.4) and other open-ended comments, it is also clear that worked examples are important for developing specific conceptual understandings (Exhibit 5). These findings are consistent with the study results of Cooper and Sweller (1987) and Trafton and Reiser (1993).

Furthermore, the open-ended comments indicated that the provision of a correct response was very helpful (

Exhibit 6). Their remarks confirm Clariana's (2000) assertion that learners should be provided with the correct response after several failed attempts, rather than continually providing multiple-try feedback throughout the entire process.

*Usability:* Overall, students agreed that it was easy to understand how to use the program (4.53). Nevertheless, one response in this area was comparatively low. When asked how well the screen layouts were designed, the response was only mediocre (3.60). A possible explanation for this result could be found in the author's field note observations, which indicated that the screen layout was cut off if a student set the monitor at the resolution of 800 x 600. In these few instances, the coding problems were not initially viewable on the screen. For example, the buttons for "submit the code" and for the feedback these students received were not shown, making it necessary for students to scroll down.

A number of other design issues were also discovered after an analysis of the field notes and open-ended comments. Because of the very long procedural steps provided within the examples, students had to keep scrolling up to see the provided output table. Also, for some students, it seemed that the long procedures were redundant, which made them inclined to skip over this portion altogether. Subsequently, many students then needed to go back to the previous examples in order to obtain the specific tags and attributes. Likewise, while studying the worked examples, seven students needed to write down the names of the tags and attributes on a sheet of paper while two students also attempted to visualize the code structure with their fingers in an attempt to map out the layout of the tables. Finally, one comment addressed the font size, which was said to impede navigation: "It was difficult to figure out how to move onto the next screen. The small red lettering in the upper right hand corner was hard to spot."

Nevertheless, many of the open-ended comments favored the Virtual Lab's instructional design, indicating that the students liked the linear presentation of examples and coding problems, rather than being allowed to choose the example or coding problem on their own: "I liked the fact that I could not move to the next screen until the correct answer was created. It made me think more about how I could make it work and enabled me to practice more."

*Utility:* Based on the students' verbal commentary during the program exploration portion of this study, five students indicated that they would like to have this type of tool in a face-to-face class or at home to practice specific programming examples. These students further asserted that their current class did not provide enough time or opportunities to practice constructing the code.

## Lessons Learned and New Research Directions

By conducting this evaluation study, I learned many lessons that have generated ideas for improving the Virtual Lab. Some of them pertained to usability issues. For example, as revealed in the data, the students had to keep scrolling up the screen to see the table shown in each example. Because of this, a picture of the output table was added on the left-hand side of every screen (Figure 3). After reviewing the students' open-ended responses, I also enlarged the font sizes of link text. Furthermore, I made critical design and development decisions for future incarnations of the Virtual Lab based on observation of learner needs and behaviors (Exhibit 7).

Over the course of the evaluation study, I also observed a number of surprising occurrences that have led me to new research directions. For instance, I found that for some students, long text-based procedures seemed to be redundant, leading them to skip forward after the first or second procedural example. These students were clearly in need of more succinct code explanations rather than step-by-step procedural descriptions. Additionally, when referring to the layout of the code, one student said, "This looks like a hamburger." I also observed two students attempting to visualize the code structure with their fingers in an attempt to map out the layout of the tables. These occurrences suggest that the students had scaffolded their thinking by creating and moving the problem space outside of the computer-based environment. These unstructured observations indicate that for some students, certain external representations—such as a hamburger—might

elicit the generation of a mental image of an object. For other students, drawing and moving the problem space externally might help them use symbols and rules correctly in generating the code. These findings have provided me with new insights regarding how the design itself could mediate the example's effectiveness on skill learning and have led me to focus on aspects of the intra-example effects, such as how an example should be designed and the way the solution is presented (Atkinson et al. 2000). Because of this, I have also conducted another study to further examine the three types of example representations: factual, procedural, and visual model examples. For a more detailed description of this follow-up study's findings, please see Kaplan and An (2005) for a review.

There is still much more that needs to be examined in order to better comprehend the effectiveness of the Virtual Lab as a learning tool. For instance, a useful avenue of inquiry might include an investigation of whether the students who were supported with the Virtual Lab might also have greater coding ability over an extended period of time. Likewise, another follow-up study could also test the immediate feedback component in the design in order to examine its direct impact on coding practice. For instance, it would be possible to compare the types of immediate feedback (i.e., knowledge of response vs. directive feedback) to discover an effective feedback mechanism for the design of this program. These findings would certainly be of interest since the provision of immediate feedback during coding practice is a critical feature of the Virtual Lab that leads to substantial cognitive and motivational benefits.

## Conclusion

In computer skills education, hands-on practice remains a crucial determinant of skilled performance (Charney, Reder, and Kusbit 1990). However, practice does not simply consist of time spent learning facts or a given sequence of steps from routine examples; it should also promote a conceptual understanding of the logic and underlying structure for such procedures. Furthermore, for novice learners, specific, accurate, and immediate feedback may be helpful in the attainment of skilled performance. Through the presentation of worked examples, the participants in this study had the opportunity to construct knowledge about coding both by following a sequence of procedures and processes and by relying on their independent hands-on experiences with problems while receiving immediate directive feedback.

It was certainly a challenge to create this first module, but it was also a learning experience. Future implementations, as discussed earlier, will continue to improve upon the initial design. As a starting point, however, I believe that the design and development of this interactive coding practice environment has the potential to scaffold and support novice learners' initial coding skill development, particularly for students attending technical community colleges in which technical proficiency is a primary concern.

## References

Atkinson, R., S. J. Derry, A. Renkl, and D. Wortham. 2000. Learning from examples: Instructional principles from the worked example research. *Review of Educational Research* 70 (2): 181-214.

Carroll, J. M. 1984. Minimalist training. *Datamation* 30 (18): 125-136.

Charney, D. H., L. M. Reder, and G. W. Kusbit. 1990. Goal setting and procedure selection in acquiring computer skills: A comparison of tutorials, problem-solving, and learner exploration. *Cognition and Instruction* 7 (4): 323-342.

Clariana, R. B. 2000. Feedback in computer-assisted learning: NETg University of Limerick lecture series. http://www.personal.psu.edu/faculty/r/b/rbc4/NETg.htm (accessed September 15, 2007).

Cooper, G., and J. Sweller. 1987. Effects of schema acquisition and rule automation on mathematical problem-solving transfer. *Journal of Educational Psychology* 79:347-362.

Kaplan, D. E., and H. An. 2005. Facts, procedures, and visual models in novices' learning of coding skills. *Journal of Computing in Higher Education* 17 (1): 43-70.

Ledgard, H. F. 1975. *Programming proverbs.* Rochell Park, NJ: Hayden.

Nielsen, M. C. 1990. The impact of informational feedback and a second attempt at practice questions on concept learning in computer-aided instruction. PhD dissertation, University of Texas at Austin.

Petre, M. 1995. Why looking isn't always seeing: Readership skills and graphical programming. *Communications of the ACM* 38 (6): 33-44.

Reed, S. K., and C. A. Bolstad. 1991. Use of examples and procedures in problem solving. *Journal of Experimental Psychology: Learning, Memory, and Cognition* 17:753-766.

Robson, C. 1993. *Real World Research: A Resource for Social Scientists and Practitioner-Researchers.* Oxford: Blackwell Publishers.

Trafton, J. G., and B. J. Reiser. 1993. The contribution of studying examples and solving problems to skill acquisition. In *Proceedings of the 15th Annual Conference of the Cognitive Science Society*, ed. M. Polson, 1017-1022. Hillsdale, NJ: Erlbaum.