Visual teaching model for introducing programming languages

Ronald Shehane
Troy University

Steven Sherman
Troy University

ABSTRACT

This study examines detailed usage of online training videos that were designed to address specific course problems that were encountered in an online computer programming course. The study presents the specifics of a programming course where training videos were used to provide students with a quick start path to learning a new programming language in their coursework. The videos addressed common learning challenges with integrated development environments (IDE) to include the following areas: structure and menu options, program code and structure, compile option, execution of program code, debugging activities, file organization and use, and output concerns. The study uses an analysis of problems encountered in using new programming languages to define the subjects that the video training should address. The study is expected to provide a guide for professional teaching practice in introducing computer programming and is expected to fill a void in research literature concerning the specific aspects of programming courses that hinder computer program learning.

Keywords: Computer Programming, Teaching Programming, Online Learning, Video Instruction, Computer Science Education

INTRODUCTION

Programming is an important part of our culture. The effects of programming are widely experienced throughout society in a multitude of social, work-related, and leisure software applications (Wright, G.A., Rich, P., & Leatham, K.R., 2012). Due to the importance and broad influence of programming, the efficient and effective teaching of the subject is worthy of study and improvement. One of the most challenging aspects of any information systems or computer science curriculum involves helping students learn the concepts of computer programming (Sengupta, 2009; Saeli et al., 2011). The task involves many complexities including skill development, different learning styles, helping students transition from a problem statement to algorithmic logic, developing programming code from that logic, debugging and resolving the program code to produce a working program that meets the problem requirements (Hadjerrouit, 2008; Taylor 2007). This learning process is even more challenging when students are also being introduced to a new programming language and development interface. Under these circumstances, learning time is often at a premium and any approach that can assist a student obtain a quick start in learning the basics of the new language and its interface give the student an advantage and help avoid initial student confusion that can hamper achieving student learning goals (Sengupta, 2009). Research findings indicate that programming students obtain their knowledge and skills from learning activities that are primarily outside the classroom, which led the authors to surmise that new approaches in pedagogy are more promising for achieving learning goals in programming (McDougall & Boyle, 2004).

Thomas Edison was quoted as saying, “It is possible to teach every branch of human knowledge with the motion picture” (Reiser, 1987, p. 11). However, since Edison’s statement, motion pictures (videos) and other digital technologies have failed to be fully utilized in teaching (Karsenti & Collin, 2011). However, research has indicated that visual approaches to teaching programming have resulted in faster comprehension and application of programming languages (Naharro-Berrocal, Pareja-Flores, Urquiza-Fuentes, & Velazquez-Iturbide, 2002). Baecker, DiGiano, & Marcus, (1997) found that visual teaching approaches were particularly effective for student comprehension of debugging tasks. Karsenti & Collin (2011) discussed the effectiveness of online videos for the professional development of teachers. The videos demonstrated live recordings of classroom activities to teacher trainees to supplement their classical teaching material. Karsenti & Collin (2011) found that the use of these videos provided practical training that enhanced teacher’s understanding and application of classroom techniques, and also provided them with the flexibility to access videos as many times as desired, any time, and from any location. Students are often hesitant to ask a teacher to repeat something more than once. However, they appear to be comfortable repeating the use of a video multiple times until they gain a full understanding of what is being presented. The availability and easy access to the videos “enabled learning through modeling and imitation” (Karsenti & Collin, 2011). Lashley (2005) found that the training videos were an effective teaching aid for technical and clinical skills. Feedback provided to Lashley (2005) indicated that the constant online availability of the videos provided students the ability to better schedule their pace of learning. Nicholson & Nicholson (2010) reported that the application of visual digital media to a course is beneficial to student learning of Information Technology (IT) material and results in improved overall student satisfaction with the learning process. The Nicholson & Nicholson study reported a reduction in the number of students requiring additional reviews and explanation of previously presented material in an IT course of study.

Research literature has demonstrated many of the advantages of using multimedia material. In particular, results from Sankey, Birch, & Gardiner (2011) indicated that students experienced improved comprehension, understanding, and retention of content. They recommended that future research might involve more complex concepts as well as an exploration of the impact on distance learning. This study was formulated to pursue those goals by examining the detailed usage of visual teaching to address specific course problems in teaching computer programming through distance learning. This study discusses an example course where a visual teaching approach was used by the authors to provide their programming students a quick start path to learning and applying new programming languages in their coursework.

The remainder of this paper is organized as follows: A discussion of the problems being encountered by the students in the course. A discussion of the visual teaching approach used to assist students in learning and applying a new programming language. A discussion of the results of using the new visual teaching approach will be presented. Following this, the conclusions and future research implications of the study will be discussed.

PROBLEM

In their statement on faculty responsibility for learning goals, AACSB places the responsibility squarely upon the shoulders of instructors in defining learning goals, assessing student's achievement of those goals, and improving their course content and approach based on their assessment. AACSB goes on to describe how a courses content needs to be continually reviewed to identify "learning experiences" that might be added to a course to enhance a student's capability to attain and sustain a course's learning goals.

A review of one of the introductory programming courses, taught by the authors, indicated that students were struggling with using the programming development interface and debugging activities on assignments early in the course which was not only affecting their efficiency in learning the new language, but represented a serious impediment to their achievement of the learning goals of the course. An analysis of the questions asked by students when compared to assessment results revealed areas of misunderstanding that students were encountering at the beginning of the course that appeared to carry forward for the remainder of the course. This was one of the root causes of student's failing to achieve the learning goals. The key problem areas that surfaced dealt with a student's ability to fully grasp the programming interface and debugging tasks early in the course. The authors had provided text and screen captured graphic presentations in previous classes that showed students how to use the interface and debugging tools of the language during the first weeks of the class. However, their analysis indicated that there was still a consistent pattern of misunderstanding indicated by student questions and assignment failures. Table 1 presents the results of the analysis that shows the problems faced by students during the early weeks of the course and examples of the types of difficulties being encountered.

SOLUTION

The authors decided to radically change the teaching approach to provide a more visually enhanced and active presentation of the language, interface, and debugging task involved. It was decided to provide archived application sharing videos, with narration, that showed the use of the programming interfaced that used complete programming examples and debugging activities. The videos were archived to enable the students to access them at any time from any location via an online course teaching platform. The tool used to create the application sharing video was Wimba Classroom Application Share using a Blackboard teaching platform. The videos were available to students to refer to as many times as desired as they performed their weekly required programming assignments. The videos were an integral part of an active learning approach that motivated programming practice and application outside normal classroom activities (Hawi, 2010).

The areas selected to be addressed by the videos were those indentified by the analysis of student questions and programming assessment shortfalls indicated in Table 1 and were as follows:

Menu Options and Structure of IDE

The narrated video began with an overview of the IDE and the various menu options available. A quick narrated overview of the various panels available in the IDE to include the editing panel and its use, compile results panel and the overall order of its use, and menu options and their overall function was discussed and highlighted on the video to help students better understand the overall order of use of the options. The overall functioning of the IDE and its purposes and why it is useful to the student was also presented to personalize the learning experience for the student and help motivate their interest.

Program Code and Structure

The video then demonstrated how to use the IDE to retrieve a programming file. The content of the program was then briefly explained as it appeared in the editing panel. The purpose of the individual commands was explained and an overview of their placement in the structure was covered. The video then demonstrated how to add additional programming commands into the overall structure using the editor. The precedence of the commands and their logical flow were also demonstrated and explained. The students were then shown how the IDE highlighted different commands and user defined variables in different colors to assist students in better understanding the structure and use of the programming code (Panell, 2003).

Compile Option

The video then demonstrated to the student the use of the compile option and the results displayed in the compiler panel. The first demonstration showed a successful compile and what the student could expect to see. Changes were then made to the program, in the editing panel, to produce a syntax error that was fully highlighted for the student and explained so they could understand an unsuccessful compile. The program was then re-compiled to show the types of error messages that students could expect in the compiler panel for different programming situations. Common types of errors were also created to show the student what they might expect to encounter during the first few weeks of programming. With each compiler error, students were shown what to do with the results and how to make corrections to the example program and recompile until workable.

Animated Execution of Program Code

The students were then shown how to execute the program in a step-by-step animated mode (Shehane, Huan, Ali, 2011). This approach permitted the student to see each command executed along with an explanation by the instructor of the flow of the program and purpose of each command. The one step execution of the entire program was also shown and how it could be efficiently used. The execution of each command was also explained in terms of the overall structure and flow of the code so that students could understand the commands within the context of the programs purpose. During the program execution video, the instructor introduced commonly encountered errors that were designed to cause the program to fail during execution to assist student in learning the difference between compiler syntax errors and execution errors. It was stressed to the students that a program can compile successfully and still not execute properly due to overall logic and flow problems in the program. This was designed to overcome a common student misunderstanding about execution failure.

Debugging Activities and Monitoring Variable

The instructor demonstrated how to debug program code with an example program in this video. A step-by-step animated execution of the program was used to demonstrate debugging. The program used was designed to reveal the variable values resulting from the execution of each command so that students could see the effect on the variable contents as each command was executed. It was then demonstrated how the IDE could be used to report of the values as the program was executed and how this could be used to identify execution logic errors, where certain commands failed to produce the expected results. This demonstration was designed to improve the overall student efficiency and effectiveness in spotting and resolving error conditions.

File Organization and Use

A video was made to show use of the IDE in saving, retrieving, and organizing programming files so that students would be better able to perform their program retrievals and submissions. The video also showed an example of each file type to include the source code, object code, and executable file and how to distinguish them in the file explorer.

Making Correction to Output and Format

The narrated video also demonstrated the use of the IDE output window to assess the suitability of the results produced by the program and the format of output reports compared with problem requirements. During this video, students were shown how to spot and correct formatting and alignment problems within the output and how to recognize the programming logic and commands that produced the various out results.

CONCLUSIONS AND FUTURE RESEARCH

The overall results of implementing the new video presentation approach of the programming IDE and debugging tools was a decrease in students programming questions during the first three weeks of the course by 72% and an improvement in students programming assessment average scores by 14% during the first three weeks of the course.

The results of the study fill a current void in research literature in terms of presenting the detailed areas of a computer programming course that can be addressed by video to resolve common misunderstandings. The study is expected to contribute to teaching practice in information systems and computer science.

A review of the results of the study indicated additional benefits and advantages resulted from introducing the new visual teaching approach to the course. The following are advantages of the teaching approach that were found:

- The video was available 24/7 for students use and Blackboard usage statistics and course access statistics indicated that students typically referred to the video multiple times during the course.
- A follow-up analysis of questions submitted in courses indicated that the problem areas addressed had successfully eliminated over 85% of the recurring questions in each category.
- A follow-up review of the quality of early programming assignments indicated that the visual teaching approach improved student’s success in developing code, free from compile errors, and free from execution problems.
- A follow-up review of the coding of early programming assignments indicated that the new teaching approach enabled students to identify compiler problem areas with their code and enhanced their ability to make incremental changes to their programs to correct the problems.

Areas for future research were identified as follows:

- The specific tool used to create the application sharing videos and the program language being taught is not expected to affect the results of this study, but this assumption should be the subject of future study.
- A rigorous student perception study, utilizing a student control group, is needed to better evaluate the usefulness of the teaching approach.

REFERENCES

AACSB. (2013) Assurance of Learning: Faculty Responsibility for Learning Goals Retrieved August 6, 2013, from http://www.aacsb.edu/accreditation/business/standards/aol/learning_goals.asp

Baecker, R., DiGiano, C., & Marcus, A. (1997). Software visualization for debugging. Communications of the ACM, 40(4), 44-54.

Hadjerrouit, S. (2008). Towards a Blended Learning Model for Teaching and Learning Computer Programming: A Case Study. Informatics in Education, 7(2), 181–210.

Hawi, N. (2010). The exploration of student-centered approaches for the improvement of learning programming in higher education. US-China Education Review, 7(9), 47-57.

Karsenti, T. & Collin, S. (2011). The impact of online teaching videos on Canadian pre-service teachers. Campus-Wide Information Systems, 28(3), 195-204.

Lashley, M. (2005). Teaching Health Assessment in the Virtual Classroom. Journal of Nursing Education, 44(8), 348-350.

McDougall, A. & Boyle, M. (2004). Student Strategies for Learning Computer Programming: Implications for Pedagogy in Informatics. Education and Information Technologies 9(2), 109–116.

Naharro-Berrocal, F., Pareja-Flores, C., Urquiza-Fuentes, J., & Velazquez-Iturbide, J. A. (2002). Approaches to comprehension-preserving graphical reduction of pro-gram visualizations. Paper presented at the Proceedings of the 2002 ACM symposium on applied computing.

Panell, C. (2003). Teaching computer programming as a language. ProQuest Education Journals Tech Directions; 62 (8), 26-27.

Reiser, R.A. (1987). “Instructional technology: a history”, in Gagne, R.M. (Ed.), Instructional Technology: Foundations, Lawrence Erlbaum Associates, Hillsdale, NJ, pp. 11-48.

Saeli, M., Perrenet, J., Jochems, W.M.G., & Zwaneveld, B. (2011). Teaching Programming in Secondary School: A Pedagogical Content Knowledge Perspective. Informatics in Education, 2011, Vol. 10, No. 1, 73–88

Sankey, M.D., Birch, D., & Gardiner, M.W. (2011). The impact of multiple representations of content using multimedia on learning outcomes across learning styles and modal preferences. International Journal of Education and Development using Information and Communication Technology 7(3), 18-35.

Sengupta, A. (2009) CFC (Comment-First-Coding) – A simple yet effective method for teaching programming to information systems students. Journal of Information Systems Education, 20(4), 393-399.

Shehane, R.F.; Huan, X.; & Ali, A. (2011).Teaching computer science courses in distance learning. Journal of Instructional Pedagogies, 6, 47-60.

Taylor, A.P.R. (2007). Programming for the Internet and experiential learning: A new approach incorporating a constructed world. International Journal of Technology and Design Education, 17(2), 217 – 229.

Wright, A., Rich, P., & Leatham, K.R. (April 2012). How Programming Fits With Technology Education Curriculum. Technology and Engineering Teacher, 3-9.

TABLE 1

Problem Analysis

Problem Areas	**Category**	**Example Problems**
Structure and Menu Options	Misunderstanding of Edit, Compile, and Results Panels	Cannot determine whether program compiled.
	Overall Use of Each Option	Changes to the program were made, but results were still the same.
	Purpose Behind IDE and Importance to Student	The order followed did not produce results.
Program Code and Structure	Program Levels	Sections of code were overlooked.
	Order of Commands	Precedence of commands was ignored.
	Color Coding of Commands	Confusion was encountered for user defined versus program code.
Compile Option	Successful Compile	Confusion about contents of compile panel.
	Unsuccessful Compile	Confusion about contents of compile panel after unsuccessful compile.
	Detecting Compile Errors	Confusion about interpreting error messages.
	Correction of Errors	Confusion concerning how to proceed.
Execution of Program Code	Execution Methods	Misunderstanding of methods of execution available and step-by-step option.
	Execution Order	Misunderstanding of order of execution within program context.
	Execution Errors	Detecting execution error versus syntax errors.
Debugging Activities	Detecting Flawed Logic	Lack of understanding of step-by-step and variable contents monitoring.
File Organization and Use	Unsuccessful Use of File Option	Failure to save files in an organized manner.
	File Type Use	Misunderstanding of different files produced including source code, object code, and executables.
Output Concerns	Output File Format	Misunderstanding of how to align columns and produce values.