AC 2010-2130: INCREASING STUDENT AND SCHOOL INTEREST IN ENGINEERING EDUCATION BY USING A HANDS-ON INQUIRY BASED PROGRAMMING CURRICULUM

Geoffrey Wright, Brigham Young University

Geoff Wright is a Professor of Technology and Engineering Education at Brigham Young University. His scholarship centers on programming, multimedia pedagogy, and technological literacy. He has published and presented on these and many other technology and engineering related topics.

Peter Rich, Brigham Young University

Peter Rich is a Professor of Instructional Psychology and Technology at Brigham Young University. His areas of focus lie in the domains of programming, design, creativity, lateral transfer, and other related topics.

Keith Leatham, Brigham Young University

Keith Leatham is a Professor of Mathematics at Brigham Young University. His areas of expertise concern advanced mathematics and computing, mathematics pedagogy, and other associated domains.

Increasing Student and School Interest in Engineering Education by Using a Hands-on Inquiry Based Programming Curriculum

First Author
Affiliation
Country
Email Address

Abstract: Many high schools nation-wide recognize the need, and are showing interest in engineering education, however, only a small percentage of those schools have been able to fully integrate an engineering component into their curriculum. The reasons for this are: lack of infrastructure, lack of training, lack of appropriate and sustainable curriculum, and lack of student interest. Paradoxically, many schools have maintained or increased the teaching of programming in their schools (Dewar, 2008). Strangely there has been little effort to correlate these two activities. Prensky (2008) stated that one of the stated core skills today's engineer need is: an understanding of computer programming. Coincidently the 2008 – 2009 employment and labor report by the U.S. Bureau of Labor Statistics predicts the need for engineers with programming experience will be one of the careers with the largest numerical increase and demand. This research outlines: 1) the need for engineering in k-12 environments, 2) analyzes the reasons for which schools have had a difficult time fully integrating engineering into school curriculum, 3) proposes a mixed content and pedagogical approach to teaching engineering and programming based on a hands-on inquiry approach, and 4) outlines additional benefits of using a blended content approach such as this (e.g., improved student mathematical self-efficacy and problem solving skills). The research project is in its second year of implementation. Last year 120 students were introduced into the course, and this year 80 more students are involved in the project. Thus far, the results of the project have shown a strong correlation between student engineering interest, aptitude, programming understanding, and an increased understanding of mathematics.

Introduction

Mathematics has long been regarded as an essential skill, as noted by the American Society for Engineering Education's mathematics division (Selingo, 2008). The Cold-War era "space race" pushed engineering awareness, mathematical, and scientific ability to the fore of our educational system. And yet, the United States exited the 20th century in a quandary over the status of its educational progress in math and science. This was due in part to the first international *Trends in Mathematics and Science Study* in 1995, which revealed that the U.S. fell behind its industrialized counterparts in advancing mathematical and scientific skills as students got older. One result was the No Child Left Behind Act (NCLB). An outcome of NCLB has been the refocusing of curriculum to allow more time-on-task for mathematics and language arts (Paris, 2000). Many districts are currently focusing their attention on more traditional classes (i.e., English, mathematics, history), reducing traditional engineering related classes, such as

technology and engineering fundamentals, applied physics, technology 1 and 2.

While the intent is to focus more heavily on fundamental language arts and mathematics understanding, recent international tests demonstrate that there has been no increase in U.S. students' mathematics scores under this new curriculum. In 2003, the U.S. participated in the Program for International Student Assessment (PISA), which tested 15-year-olds' science and math skills, placing above average internationally in both categories. Three years later, on this same test, U.S. students' scores were statistically identical, but they were outperformed by 16 other industrialized nations in science, and by 23 nations in mathematics (only 30 nations participated). Narrowing the curriculum is not advancing the U.S.'s educational system and is inadequately preparing students to compete in a 21st century world.

Lateral Transfer

Rather than reduce the curricula, research indicates that systematically pairing specific subjects may improve both learning and motivation. For example, research consistently demonstrates a strong correlation between second language (L2) learning and increased first language ability on standardized achievement tests. L2 learners have greater: syntactic awareness (Bialystock, 1988, Galambos & Goldin-Meadow); phonological awareness (Bruck & Genesse, 1995; Campbell & Sais, 1995); and concept of word labels (Bialystock, 1988, Ricciardelli, 1992). Additionally, one of the strongest positive predictors of successful performance in computer science programs is prior math experience and achievement (Bergin & Riley, 2006). The focus of this study is on this latter correlation between mathematics and computer programming.

Predictors of Success in Computer Science Courses. A myriad of research investigates predictive success factors in Computer Science courses. Multilinear regression models generated in this research overwhelmingly include prior math experience and achievement. This may involve high school math achievement scores, math scores on nationally recognized tests (e.g., the SAT), and enrollment in math courses. In one study, Leeper & Silver (1982) used data from first-time computer science students' SAT scores (verbal and math), high school class ranking, and high school grades in Math, Eng, Language Arts, and Science and compared these to their course scores. A multivariate regression analysis revealed that over half of the model's variance (26%) was attributed to mathematics (14.3%). Campbell & McCabe (1984) used Multivariate regression analysis, Chi-square tests, and Discriminate analysis to construct and compare student attributes. The difference between those who stayed and those that left, "were related to the students' SAT math and verbal scores, their high-school rank, and their background in high-school mathematics and science" (p. 1113). This is similar to Konvalina, Wileman and Stephens' (1983) finding that 228 students that completed an introductory computer science course had significantly more mathematics background than the 153 that withdrew.

Mathematics is always considered in these models because, "there is a belief that the concepts which a student has to comprehend in order to master mathematics problems are similar to those for programming. Mathematics aptitude is thus often a pre-requisite for acceptance into computer science" (Bryne & Lyons 2001, p. 51). The corollary also appears to be true, that learning to program enhances one's mathematical ability. The National Mathematics Advisory Panel recently issued the following statement in "Foundations for Success: Final Report" (2008):

> The Panel recommends that computer programming be considered as an effective

> tool…for developing specific mathematics concepts and applications, and mathematical problem-solving abilities. Effects are larger if the computer programming language is designed for learning (e.g., Logo) and if students' programming is carefully guided by teachers so as to explicitly teach students to achieve specific math goals. (NCTM, 2008, p. 52)

The purpose of this study is to examine the synergistic relationship between mathematics and computer programming, when taught to high school students.

The Study

We are currently in our second iteration of the research study. During year 1 we developed the curriculum that would be taught in the programming classes – however, it has since continued to evolve and be modified, additionally we developed the pre and post mathematical and programming assessments. Finally, we performed a pilot study where two classes of 7th and 8th grade students were taught the programming curriculum. The classes consist of a primarily middle to upper class Caucasian male and female students. Adobe Flash was used as the programming interface, because it was one of the software provided and approved by the school district. Additionally we felt the software provided a friendly and easy to use interface, and provided the students with a tool that would enable them to quickly create observable projects (many of the existing programming tools are not very intuitive, do not provide the code hinting and feedback that Flash has built in, nor do they allow for students to observe what they program in a visual format). Also, Flash was the decided tool for the curriculum and study because the new ActionScript that Flash uses is based on Java, where variables, functions, methods, events, properties, algorithms, and other fundamental and essential programming principles that are related to mathematics are used. During the pilot study the teacher was tutored and assisted by two undergraduate technology and engineering education pre-service teachers. The pre-service teachers supported the teacher in his effort to integrate the new curriculum into his existing curriculum, and quickly learn the software and basics of programming. Involving the pre-service technology and education majors had a secondary benefit, because it provided the pre-service teachers the opportunity to be involved in an actual real teaching environment where they were helping develop curriculum, lesson plans, interacting with students, and so forth. During the pilot test the two classes were first given a mathematics pre-test that asked them questions about variables, use of operators, functions, coordinates, and equations. The students were then taught the GUI (Graphical User Interface) of Flash, and by the second day of class were being taught the basics of programming. The curriculum was based on a game design pedagogy, where students were taught to program by having them create basic games in Flash. For example during the first week the students were taught about variables by having them create a simple Zelda© type game where they had to program a character/avatar to walk left and right using the arrow keys while avoiding moving objects. Each of the games were given to the students with certain lines of code missing, and they were asked to complete the code and make the game work. The code they were to fill in centered on the principle being taught that day. For example, later in the semester the students were asked to make a matching puzzle game just after learning about mathematical operators and Boolean expressions. The capstone activity for the class required the students to design and program their own game – based on the content they had learned during the course of the semester. At the end of the semester students were asked to take an exit exam testing both their programming and mathematical understanding, titled the "Mathematical problem-solving inventory". The math portion of the exit exam/inventory was very similar to the

pre-test administered at the beginning of the semester. The mathematical problem-solving inventory was also administered to 30 other students at the same school of the same grade level who did not participate in the class. The mathematical problem-solving inventory was based on Knuth's (2005) and Weinburg's (2004) work investigating student mathematical growth.

Procedures and Methodology

There are 3 phases to this research project: (a) establishing a baseline, (b) administering the intervention, and (c) measuring possible effects. We discuss each in turn.

Establishing a baseline

To establish a baseline, we used two measures. 1) We gave the mathematical problem-solving inventory to both students enrolled and not enrolled in the programming class. The students who were not enrolled in the programming class and who took the mathematical problem-solving inventory served as our control group, whereas the students who were enrolled in the programming class and who took the mathematical problem-solving inventory were our experimental group. The comparison of the two groups served as one part of helping establishing a baseline measure. 2) The second baseline measure was based on the students standardized test scores as retrieved from the district-level student information system. This system tracks student progress on standardized tests (e.g., the Criterion Referenced Test). These test scores were compared with a second randomly selected control group to assure academic equivalence. Academic equivalence will be determined using prior standardized math scores as reported by COGNOS.

Administering the intervention

Experimental group students were taught by the in-service teacher of record. Two undergraduate technology and engineering education pre-service teachers worked as co-instructors for roughly 60% of the courses, assisting in the teaching of concepts thought to be shared between mathematics and programming, such as variables, functions, parameters and problem solving. While the course focused on learning common programming aspects, the curriculum was designed to focus more readily on core programming principles, such as methods, behaviors, classes, prototypes, etc. In order to accomplish these ends, the curriculum and instructional activities used a game-creation approach, wherein the students developed increasingly complex games to implement said programming concepts.

Measuring possible effects

Following the conclusion of the course, control and experimental students took an alternate form of the mathematical inventory taken prior to the course. In addition to the mathematics inventory, researchers interviewed experimental group participants to gauge to what extent participation in the course may have affected their: (a) interest in math, (b) mathematics self-efficacy, (c) interest in computer programming, (d) anticipated future engineering programming activity, and (e) unexpected outcomes of participating in the course. In these interviews, researchers encouraged participants to discuss specific projects they designed and the process they went through to complete these designs. The data analysis consisted of a pseudo mixed methods approach. At present time we have recently finalized the codes to use in analyzing this past year's student responses to the mathematical inventory. Initially we anticipated using Knuth's (2005) and Weinburg's (2004) coding methods to analyze the inventory, but during our first weeks of

analysis we learned the codes did not provide sufficient interpretation of the questions we were seeking. Consequently we further investigated possible coding methods and terminology. Ultimately we developed our own coding system, which we are currently using to analyze the data. It is anticipated by the ASEE 2010 conference we will have sufficient data analyzed, from which we will be able to provide further details on the impact this study is having on lateral transfer. Additionally, we feel this year's iteration of the research project will better inform the pilot study's data because several modifications were made to the pre and post mathematical inventories, survey, and curriculum.

Findings and Conclusions

As stated above we are currently in our first data analysis phase analyzing pilot study data points, and are commencing the second iteration of data collection. However, we believe the results from this research will ultimately have a direct impact on high school students' conceptual understanding of various mathematical functions and processes, increase their processing skills by exposing them to various dynamic programming activities that will push their problem solving abilities, require them to solve out-of-content problems, and be more creative. Exposing high school students to programming at an earlier age in the type of programming environment described in this proposal will benefit them by helping them better understand mathematical contexts, concepts, and applications, often reflected in applied fields such as engineering.

Bibliography

1. Austin, H. S. (1987). Predictors of pascal programming achievement for community college students. Proceedings of the Eighteenth SIGCSE Technical Symposium on Computer Science Education, , 161-164.

2. Bergin, S., & Reilly, R. (2006). Predicting introductory programming performance: A multi-institutional multivariate study. Computer Science Education, 16(4), 303-323.

3. Bialystock, E. (1988). Levels of bilingualism and levels of linguistic awareness. Developmental Psychology, 24, 560-567.

4. Bialystock, E. (2001). Bilingualism in development: Language, literacy, and cognition. Port Chester, NY, USA: Cambridge University Press.

5. Bruck, M., & Geneese, F. (1995). Phonological awareness in young second language learners. Journal of Child Language, 22, 307-324.

6. Byrne, P., & Lyons, G. (2001). The effect of student attributes on success in programming. ACM SIGCSE Bulletin, 33(3), 49-52.

7. Campbell, P. F., & McCabe, G. P. (1984). Predicting the success of freshmen in a computer science major. Communications of the ACM, 27(11), 1108-1113.

8. Campbell, R., & Sais, E. (1995). Accelerated metalinguistic (phonological) awareness in bilingual children. British Journal of Developmental Psychology, 13, 61-68.

9. Friel, B. (2003). Don't know much about history. National Journal, 35(31), 2550-2551.

10. Galambos, S. J., & Goldin-Meadow, S. (1990). The effects of learning two languages on levels of metalinguistic awareness. Cognition, 34, 1-56.

11. Honour-Werth, L. (1986). Predicting student performance in a beginning computer science class. Proceedings of the Seventeenth SIGCSE Technical Symposium on Computer Science Education, , 138-143.

12. Hostetler, T. R. (1983). Predicting student success in an introductory programming course. ACM SIGCSE Bulletin, 15(3), 40-43.

13. Knuth, E.. Alibali, M. W., Weinberg, A., McNeil, N., & Stephens, A. (2005). Middle school students' understanding of core algebraic concepts: Equality & variable. Zentralblatt für Didaktik der Mathematik (International Reviews on Mathematical Education), 37, 68-76.

14. Konvalina, J., Wileman, S. A., & Stephens, L. J. (1983). Math proficiency: A key to success for computer science students. Commun ACM, 26(5), 377-382.

15. Leeper, R. R., & Silver, J. L. (1982). Predicting success in a first programming course. Proceedings of the Thirteenth SIGCSE Technical Symposium on Computer Science Education, , 147-150.

16. Paris, S. G., & McEvoy, A. P. .. (2000). Harmful and enduring effects of high-stakes testing. Issues in Education, 6(1/2), 145-160.

17. Ricciardelli, L. A. (1992). Bilingualism and cognitive development in relation to threshold theory. Journal of Psycholinguistic Research, 21, 301-316.

18. Rosenthal, B. (2004). No subject left behind? Think again. NEA Today, 23(2), 26-27.

19. Selingo, J. (2008). Game of Chance. Journal of American Society for Engineering Education, 17(5).

20. Stewart, J. H. (2005). Foreign language study in elementary schools: Benefits and implications for achievement in reading and math. Early Childhood Education Journal, 33(1), 11-16.

21. Weinberg, A., Stephens, A. C., McNeil, N. M., Krill, D. E., Knuth, E. J., & Alibali, M. W. (2004, April). Students' initial and developing conceptions of variables. Paper presented at the Annual Meeting of the American Education Research Association, San Diego, CA.

22. Wilson, B. C., & Shrock, S. (2001). Contributing to success in an introductory computer science course: A study of twelve factors. 33(1), 184 - 188