Scratch 2 as an Introduction to Programming: Does it improve knowledge of programming methods?

Timothy L. Smith
LTEC 5610

Abstract

As systems and machines traditionally operated by manual human labor have become increasingly automated, many opportunities for careers in engineering and computer science have developed. More and more common items are being designed to be connected to networks and controlled by computers, and with everything from fast food cashiers to truck drivers being systematically replaced by machines, the demand for workers in these fields will continue to increase. With positive job growth and increasing options for computer science degrees from many universities, the introduction of programming concepts to children is becoming a crucial component to their formal education. In this proposed study, traditional programming instruction will be compared to a visual based programming language that emphasizes important concepts rather than rote memorization of syntax.

Scratch 2 as an Introduction to Programming: Does it improve knowledge of programming methods?

It is easy to assume that younger populations are inherently adept in all forms of technology because they were born into an age of mobile computing and embedded devices. This type of stereotyping can be dangerous, as it ultimately limits the potential of an entire generation. Although most students in K-12 education utilize computers and tablets during their curriculum many are not properly introduced to computer science and may only have a limited understanding of how their devices work. By introducing them to programming concepts through an accessible visual language such as Scratch during their formal education, their knowledge of concepts and interest in the field should rise along with their intention to seek a computer science degree during their higher education experience and ultimately their career.

Literature Review

With computing education becoming more of a focus for program administrators and instructors, it is imperative that computer science researchers identify critical needs and create an environment of learning and retention for an audience of diverse learners. Skills in computing are nearly essential to 21st century employees, but only a small portion of students are introduced to necessary computer science concepts. Furthermore, the students who do study computer science are not typically diverse. As US adoption of computer science before higher education expands, it is important that initiatives to engage students who are underrepresented becomes fundamental (Buffum, Frankosky, Boyer, Wiebe, Mott & Lester, 2016). One way to increase engagement and capture a variety of learners is to implement collaborative game-based learning. In the study carried out by Buffum et al, a game with a similar interface to Scratch was utilized in place of traditional programming methods to introduce challenging concepts such as *broadcasting*. This programming concept refers to sending a message from one device or sprite to another device or sprite that is listening for the message. When the message is received a script is triggered to run. This concept is particularly challenging to K-12 students, but the use of this platform combined with collaborative pairings resulted in a marked improvement (Buffum et al., 2016).

Another issue plaguing STEM students is the lack of visible role models to boost self-efficacy. Without role models that are relatable, the students do not see evidence of success, hence they see problems as unsolvable (Fernandez, 2013). The combination of relatable role models and recognizable methods could be the key to success. If more familiar methods are used to present these seemingly insurmountable concepts, it stands to reason that improvements in performance and diversity would be seen in the field. Game based learning, visual programming languages, and building block style coding are all great examples of ways to familiarize diverse, underrepresented populations with computer science.

A challenge for educators in a field that is rapidly encroaching on K-12 curricula, is that of transferring knowledge in a highly technical subject with expectations of advanced concept mastery. Adopting the top-down approach utilized in higher-education

is not effective, and often results in confused students who never further their education in the field. This problem is already concerning in a course that is generally offered as an elective, but it will be even more concerning when many states adopt computer science as a requirement for K-12 students (Knobelsdorf and Vahrenhold, 2013). In the journal article by Knobelsdorf and Vahrenhold alternative methods such as the Spiral Curriculum are proposed. Instead of direct transfer of college computer science to K-12, only topics that are worthwhile when fully developed should be taught. The Spiral Curriculum also calls for repeatedly teaching topics while increasing complexity over grade levels. This aligns well with the study proposed in this paper where a non-traditional form of programming is introduced in the ninth grade and can be expanded upon at later grade levels.

The final initiative that should be mentioned in relation to the study is the Hour of Code event. In this worldwide project, the goal is to provide greater accessibility and to encourage computing skills, logical thinking and critical thinking skills among diverse and underrepresented populations. The event takes place during the Computer Science Education Week each year. One instance in which 100 students (ages 6 to 46) participated in the Hour of Code, many of the participants wrote their first computer applications. Later a similar program was organized on a local level, and students from K-8 were excited to "play computer games" when in actuality they were learning programming concepts (Bojic, Jagust & Sovic, 2015).

Methods and Procedures

The proposed method would take two classes chosen according to the sample below. Both classes will be provided with a similar course of study in relation to programming concepts, but one class will use traditional textual code to produce projects. The other class will utilize a visual programming language known as Scratch produced by MIT. Again, the concepts and resulting projects should be the same throughout the course of study, but the programming language and input methodology will differ.

The two classes of ninth grade students will be administered a pre-test to gauge their knowledge of computer software and programming. This will provide a baseline to compare to once the course has ended. It is expected that regardless of group (control or treatment) the knowledge levels should increase, but it is unknown if one method will outweigh the other. The students will also be given a pre-survey to determine their interest in the field, demographic information, and their intentions after high school graduation. This pre-survey will be used to measure student engagement and higher education plans per student. The same question set will be used for both groups with the pre-test being multiple choice and the pre-survey being Likert-scale format.

Then throughout the semester each class will be taught the same basic concepts such as sequencing, loops, variables, selection and Boolean operators. One of the classes will be taught programming concepts via a traditional written language such as Python, and the other will be taught using MIT's Scratch visual programming language. The aforementioned concepts will also be tied to projects such as games, applications, and animations. Again, the students in the control group will develop these projects using the

Python language, while the treatment group will use the more visual Scratch building blocks.

At the end of the semester both groups will be administered the same version of a post-test to determine their knowledge level after the study. The test will contain a combination of traditional multiple-choice questions and scaled self-assessment questions. These questions will be compared to the pre-test results to determine which course of study had the greatest variance. The students will also be given a post-survey to gauge whether or not their interests and intentions have changed. It is expected that interests will be higher among the treatment group, and plans to enroll in computer science or STEM related programs will increase.

Research Questions

Will the introduction of Scratch alongside programing concepts to ninth grade students increase their understanding of object-oriented programing? Would a program of study that includes visual programming courses correlate with a rise in computer science majors?

Sample

The sample for the proposed study would be two ninth grade classes from school districts with similar standardized test scores and funding. Both schools would need to already have a computer science elective in place, with the control group remaining on the traditional course of study. The treatment group would follow a new course of study that includes applications and concepts similar to the ones detailed in the literature review. This sample would be quasi-random with the only conveniences being related to region and income level.

Type of Design:

The design is quasi-experimental with a somewhat convenient sample. The test retest method will be used to determine effect size between control and treatment.

Independent Variables:

The use of Scratch programming software as an alternative to a more traditional programming environment.

Dependent Variables:

The results of the pre-tests, pre-surveys, post-tests, and post-surveys will all be utilized to determine the outcome of the predictor.

Data Analysis

Initially data from the pre/post tests and surveys from the control group and treatment group would be examined in isolation. The two pools of data would be analyzed separately via frequency charts, looking for distributions. Then the data sets could be compared to measure for skewness. Theoretically the treatment population should have a higher mean, proving efficacy. Examining specific traits would provide further analysis to these initial observations. If the data looks similar between the two populations, it should be further analyzed. By looking for outliers and examining specific traits within overlapping quartiles, it could be determined if characteristics of one quartile were elevated compared to another. For example, if a group was in an upper quartile in the control group but dropped to a lower quartile, it may show that the approach has adverse effects with certain populations.

Results and Significance

The results of the study should prove or disprove the effectiveness of implementing alternative programs of study in the field of computer science among K-12 students. This is significant due to shifts occurring in the job market and society alike. Computing has touched nearly every aspect of 21st century life, and preparing students for the inevitability of working in an industry saturated with computing technology is paramount. According to Dovi, an instructor of computer science in Richmond, Virginia, "Computer science in society has shifted significantly. It is no longer a skill that a small set of people need. To succeed in a much wider range of career areas, students now need a deep understanding of computing." (Stephenson & Dovi, 2013, p. 44)

References

Bojic, I., Jagust, T., & Sovic, A. (2015). Selected examples of cooperation between universities and schools in STEM education. *2015 IEEE Integrated STEM Education Conference*. doi:10.1109/isecon.2015.7119921

Buffum, P. S., Frankosky, M., Boyer, K. E., Wiebe, E., Mott, B., & Lester, J. (2015). Leveraging collaboration to improve gender equity in a game-based learning environment for middle school computer science. *2015 Research in Equity and Sustained Participation in Engineering, Computing, and Technology (RESPECT)*. doi:10.1109/respect.2015.7296496

Fernandez, M. (2013). A Path Between: Mentoring the Next Generation of Computing Professionals. *Computer,46*(3), 38-41. doi:10.1109/mc.2012.396

Knobelsdorf, M., & Vahrenhold, J. (2013). Addressing the Full Range of Students: Challenges in K-12 Computer Science Education. *Computer,46*(9), 32-37. doi:10.1109/mc.2013.263

Stephenson, C., & Dovi, R. (2013). More than Gender: Taking a Systemic Approach to Improving K-12 Computer Science Education. *Computer,46*(3), 42-46. doi:10.1109/mc.2013.2