TJHSST Computer Systems Lab Senior Research Project
Computer Science for the Young Mind

Paul Im
2008-2009
Alexandria, Virginia

March 31, 2009

Abstract

Technology has made tremendous leaps and bounds throughout the ages. More and more of todays work force has taken to the field of computer programming. Sadly enough, there has been limited effort to teach computer science at the elementary school level, but progress is being made. The purpose of this project is to implement computer programming to educate elementary school students in math and science. Hopefully, this class will last long after we stop teaching and, along with other programs like it, will become an inspiration to all who teach computer programming.

Keywords: Computer programming, elementary, math, science

1 Introduction

Since the beginning of civilization, mankind has made momentous leaps and bounds in technological advancement. From the Industrial Revolution of the 1880s to the dawn of the Digital age, because of how far weve come, we now live in a world once deemed unimaginable. At the center of this new world sits an adapting education system, where more and more people are taking up

jobs in the field of computer science. Unfortunately, very little progress has been made at the lowest level of education in terms of technology: children.

Although there has been some progress in the past, theres still a lack of significant progress on a large enough scale to be considered great. The first attempt to educate elementary school students was with Logo, a programming language that used a turtle sprite to draw figures with a pen. Soon, other programming languages followed, each one improving on the last. Still, computer programming education is a major issue in the world when it comes to students at the elementary school level.

Just how young is too young to start programming? The purpose of this project is to answer this question by advancing an already successful computer science program at Cardinal Forest Elementary School via Scratch, a programming language developed by MIT. (Gates, 2008) Along with fellow students Jessica Gorman and Crystal Noel, I helped teach the students. Though not all the students at Cardinal Forest Elementary participated, enough of them did so to sustain the program.

2 Background

2.1 Computers, Children, and Education

Traditional computer science programs utilize complicated programming languages, such as Java, Python, or C++, all of which are geared toward high school and college students. The first attempt to teach computer programming to younger children was with Logo, which involved telling a turtle how to move around to make various pictures. Since then, other preliminary programming languages, including Squeak, Alice, and Scratch, have been implemented with varied levels of success.

The necessary technology to teach Elementary students computer programming does exist, but unfortunately, most computers are used to reinforce allegedly outdated teaching methods, most commonly as a medium for transferring information. This method has been proven to be very ineffective, possibly due to a level of educational ineptitude on part of the students being left with little to do. As several studies have shown, students learn better when they immerse themselves in lessons instead of simply listening to lectures (Gates, 2008).

The goal of this project was to continue the development of an ongoing

computer science curriculum at Cardinal Forest Elementary School. First started in 2007 by Gregory Gates, the curriculum, inspired by "computer clubhouses" at the Massachusetts Institute of Technology (MIT), consisted of weekly sessions that run from 30 to 45 minutes at a time. Not all students participated, but those that did were integral parts of the project.

2.2 Scratch

Scratch was developed and released in 2007 by MIT; it gets its name from its dynamic editing style, allowing users to edit programs as theyre still running, similarly to how DJs mix music during performances. Instead of typing commands on various lines of code, the programming language editor uses simple, colored boxes and images to provide coding, as if assembling structures with building blocks (Fildes, 2007).

Reminiscent of art programs such as Kid Pix, the colorful and intuitive interface makes it easy for users to tell that Scratch was specifically geared toward a younger audience, and users need only to drag boxes to designated programming fields and connect them together as they see fit. Users are even given the ability to create and edit custom sprites. Furthermore, its influence isnt limited to children; at Harvard University Extension School, it has been tried out in some introductory computer science classes as a means for students to start making solid programs early (Johnson, 2007).

As of November 2008, the latest release is version 1.3.1, compatible with Windows and Mac operating systems. An unofficial version of Scratch 1.2 is available for Linux.

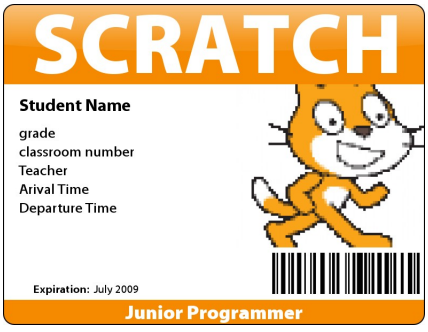

Figure 1: This image was used as a sample student badge that students needed to display to Mr. Allard in order to participate.

3 Procedures and Methods

3.1 Timeline

In September, I heard of the possibility of implementing a computer science program for elementary school kids and took up the opportunity after careful consideration. I contacted the principals of elementary and middle schools nearby Thomas Jefferson High School inquiring about such a possibility. After weeks of hearing nothing from said principals, I received a call from Mr. Frederic Allard, the same teacher contacted by Gregory Gates last year.

We went to work shortly after I confirmed my willingness to participate. Also working on the project were Crystal Noel and Jessica Gorman, who had previously been chosen as successors to Gates' program, but were nonetheless more than happy to include me. Soon afterwards, we met together in the library to discuss who would cover which aspect of the program; Crystal would investigate Scratch as an online community, Jessica would determine whether or not grade level is a factor in learning capabilities, and I would see how effective unconventional teaching methods are.

The program had to be expanded on; my teammates and I had to work out how we would divide students who had previously taken the Cardinal Forest Elementary School Scratch course from those who had not. The three of us made different contributions to help teach the children and studied various aspects of teaching in the process; by October 9th, 2008, the class had started.

After spending the first quarter using Scratch as a teaching tool for both mathematics and computer science, Mr. Allard introduced various projects for the students to apply their knowledge to. One of which was a "Kitty Rectangle" challenge for the children to work on until December. The goal of this project was to incorporate mathematics into technology; more specifically, it taught the students the concepts of area, perimeter, and angles of a rectangle. Later, the students worked on a Winter Wonderland project, whereupon they had to make a snowman move around an ice skating rink indefinitely by using loops in the programming language.

3.2 Class Structure

As with last year's program, the class wasn't so much a class as it was a computer club. Every Thursday, the students met with Mr. Allard for lessons

designed in part by Jessica, Crystal, and me. Students signed in (for attendance purposes) and quietly sat down at their computers, and then teaching began. We did, however, tweak the program a bit to further accommodate for the various age groups and levels of experience represented in the class, which was divided into five sessions. Each one housed a different grade level and assembled at a different time of day. For example, the kindergartners and fifth graders met in session D, between 1:15 PM and 1:55 PM. Also, starting October 16th, 2008, students were given special badges to distinguish themselves from other students.

I was the main program writer of the group. Every week or so, I would create and submit a movie made with Scratch for Mr. Allard to review, edit, and post on Blackboard, all of which were used as teaching aids. Since there was no direct way of knowing whether or not the children were learning anything, I just monitored the progress made by the students in each session. Depending on how good or how bad progress was, I made adjustments to my style of programming.

Eventually, I had to see the students face to face via video conferencing to assist them more directly. This initially brought on numerous complications, such as proper preparation of equipment and lack of a feasible means of transportation, but were later addressed. By the end of March, we started conducting video conferencing sessions which, despite some initial technical difficulties, allowed me to more readily gain a sense of how well the students were learning.

3.3 Topics

As with last year's program, the number of topics we covered was dictated by how quickly the students could move from one topic to another. , so we often used Scratch to directly teach the students about angles, integers, axes, etc. Overall, we primarily focused on linking science and mathematics to each other; in order to navigate around the program's output window, students needed to understand the coordinate system and basic geometry. We further emphasized the need for mathematical knowledge in the projects we designed, as well as varying features about Scratch. Ranging from a challenge about the dimensions and properties of a rectangle to a movie of a snowman circling an ice skating rink, each one incorporated a different aspect of each subject. While we couldn't teach everything there was to know about Scratch, we did what we could and hoped the students would figure out certain aspects on

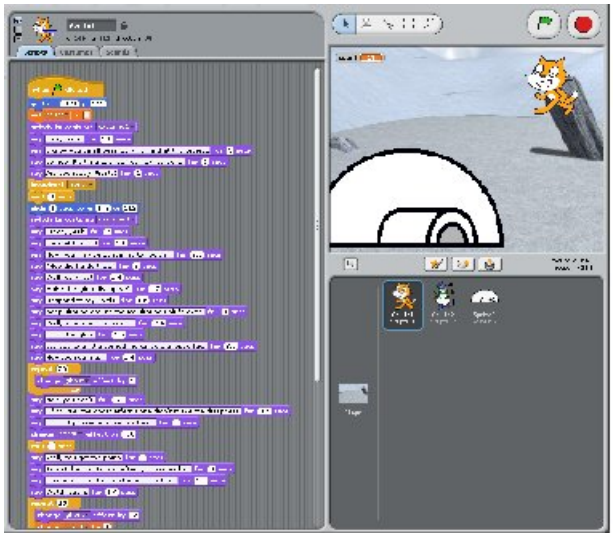

Figure 2: This program was used as a project demonstration for the students in January.

their own.

The potential of Scratch as a teaching aid is not limited to mathematics and technology. The coding is designed in such a way as to be almost like pieces of a colorful puzzle, so in a sense, not only is a user programing, but also painting a picture. The projects we designed allowed the students to apply what they learned from Jessica, Crystal, me, and Mr. Allard in class and at home. In addition, the presentations scheduled into the curriculum allow students to practice and develop their language arts skills, some more so than others.

4 Results and Conclusion

4.1 Expectations and Aspirations

As of December 2008, I'm hopeful that all will go according to plan. The students will hopefully learn Scratch efficiently and have a wonderful time doing so. I'll manage to get the project done well and aid the students effectively, even though I may never see their faces for myself. With the success of a WHUT broadcast documenting the Scratch program at Cardinal

Forest, I think I really made a difference. If I finally get the chance to see the kids for myself, who knows? Maybe they'll want to talk to me about how hard it was to make all those movies, or something.

If the students aren't learning as well as they should, I'll have to contribute to revisions in the curriculum, such as restructuring the format of the Scratch movies I periodically post on Blackboard or suggesting alterations in the teaching structure. Hopefully, the consequences won't be so severe as to cause the whole curriculum to fall apart.

4.2 Results

As of March 2009, the program is struggling a bit, but is still a success. The students appear to be well on their way to learning what they need to learn, and weve made adjustments to the curriculum as necessary. I still see room for improvement, however; the lessons were, on occasion, poorly coordinated due to lack of communication between Jessica, Crystal, and me. As such, there have been widening differences in how we each approached the curriculum, ranging from teaching methods to lessons. Fortunately, theyve all led to similar levels of success.

4.3 Discussion

Contrary to what many people seem to believe, computer programming can be–and has been–taught to students at the elementary school level, albeit through radically different means than one would normally expect. The earlier they start programming and showing interest in computers, the better. In the end, the students have made great progress in familiarizing themselves with programming through Scratch, and from what I've seen, someday, the computer may very well become the new medium for teaching nearly all subjects, including English and Social Studies. So, how young is to young to start teaching kids how to program (Gates, 2008)? Who knows? Maybe there's no such thing as "too young" to do so.

References

[1] Feldman, Michael B., and Bruce D. Bachus, *Concurrent Programming CAN be Introduced into the Lower-Level Undergraduate*

Curriculum, Ms. 16 Sept. 2008. ¡http://www.seas.gwu.edu/ ada-group/concurrency/bachus/¿

[2] Fildes, Jonathan, "Free Tool Offers 'Easy' Coding.", BBC 14 May 2007. 6 Oct. 2008 ¡http://news.bbc.co.uk/1/hi/technology/6647011.stm?ls¿

[3] Gates, Gregory, *TJHSST Computer Systems Lab Senior Research Project Paper; Elementary Education in a Technology Age*, Ts.

[4] Hazzan, Orit, Judith Gal-Ezer, and Lenore Blum, "A Model for High School Computer Education: The Four Key Elements that Make It", ACM SIGCSE Bulletin, 40.1 (2008): 281-285.

[5] Helmut Kopka and Patrick W. Daly, A Guide to LATEX, Addison-Wesley Publishing Co., Inc., 1993.

[6] Hoffman, Mark E., Timothy Dansdill, and David S. Herscovici, "Bridging Writing To Learn and Writing in the Discipline in Computer Science Education" ACM SIGCSE Bulletin, 38.1 (2006): 117-121.

[7] Maloney, J., Burd, L., Kafai, Y., Rusk, N., Silverman, B., and Resnick, M. (2004). "Scratch: A Sneak Preview." *Second International Conference on Creating, Connecting, and Collaborating through Computing.* Kyoto, Japan, pp. 104-109.