TJHSST Computer Systems Lab Senior Research Project Paper
Elementary Education in a Technology Age
2007-2008

Gregory Gates

6/10/08

Abstract

Technology becomes more advanced and more accessible with every passing day. Education should be utilizing this technology boom in teaching current students. However, this does not seem to be the case. The goal of this project is to try and implement computer programming, through Scratch, as a tool for educating students in math and science topics. This type of computer science education at a younger age is increasingly beneficial as computers become a more prominent facet of life.

1 Introduction

The main question that this research project aims to answer is, "How young is too young to start teaching children how to program?" The goal is to establish a computer science program at Cardinal Forest Elementary School through the use of the MIT developed program 'Scratch.' The plan was to watch how the students used the program so that I could discover an answer to this question.

This paper will detail both how Mr. Allard and myself taught the students, and how the students went about trying to solve the problems that they were presented with. In order to ensure this project was more than

just the musings of a teenage scientist, Mr. Allard and I correlated our lessons with the Virginia Standards of Learning (SOL) and Program of Studies (POS) benchmarks for the students. The children that participated in the program were in kindergarten through sixth grade, with each grade having a similar curriculum. Not all of the elementary school students participated in this project however. Mr. Allard selected a diverse group of students for this initiative that he thought would do best in and benefit from a program such as this.

2 Background

2.1 Children and Programming

The task of educating the younger generations about programming has been attempted before. The first attempt to create a kid-friendly programming language was Logo, made by Wally Feurzeig and Seymour Papert. This programming language mainly involved telling a turtle how to move around in order to make various pictures with the turtle's "pen." Since then, multiple programming environments and languages have come about to try and engage not only youth but more specifically girls in computer science and programming such as: Squeak, Alice, and Scratch.

Despite the bountiful number of tools that modern technology gives us for teaching students, little progress has been made in teaching computer science at the elementary school level. The necessary technology is present in the schools, but it is only being used to reinforce outdated teaching methods. Currently, computers are mainly being used as a medium to transfer information, much like a television. Computers have so much more potential than that. They should be used as a universal construction material, not as a TV screen. Programs like Scratch enable kids to create whatever they want to all by themselves. Children learn better by immersing themselves in whatever they're doing, rather than just listening to a teacher telling them what to do (Papert, 1993).

The goal for this project is to establish something akin to a Computer Clubhouse at Cardinal Forest Elementary School. The original Computer Clubhouse was started by the Massachusetts Institute of Technology in Boston in 1993 to "provide more young people with the opportunity to become digitally fluent." (Resnick, 2002) Mitchell Resnick describes these clubhouses

as a place where kids and older youth "become designers and creators with new digital technologies. Clubhouse members use leading-edge software to create their own artwork, animations, simulations, multimedia presentation, musical compositions, websites, and robotic constructions" (Resnick, 2002). These clubhouses have been an important inspiration for this project.

2.2 Scratch

Scratch is essentially a visually based programming language that is specifically geared to students between the ages of eight and sixteen. Scratch got it's name from the scratching technique DJ's use when mixing music during a performance (Johnson, 2007), and this inspiration can be seen throughout the program. People that use Scratch are able to take the different, colored, simple blocks of code and put them together into a single creative product much like DJ's do when when mixing two songs together. One can tell from the start that Scratch's colorful and intuitive interface is definitely geared towards a younger audience, however the potential age group of people that could learn programming using Scratch is limitless. Some colleges are actually using Scratch in their introductory computer science courses (Johnson, 2007). Scratch is a powerful tool that allows skills like creative thinking and systems analysis to be in the same program with some outstanding results.

3 Development Sections

3.1 Timeline

This past fall, I contacted the principals of the middle and elementary schools in the West Springfield pyramid in Fairfax County, Virginia inquiring about the possibility of starting a computer science program at their schools. Only one school replied (Cardinal Forest) and the principal referred me to Cardinal Forest's School Based Technology Specialist, Mr. Frederic Allard. October and November were then spent sorting out which programming language to use and how we were going to use it. After experimenting with 'Squeak' and 'Alice' (developed by Carnegie Mellon) we settled on 'Scratch.' It was decided that the Scratch students would meet for the program during their recess time every Thursday. I personally teach the students every other Thursday, with Mr. Allard teaching on the days that I can't make it. The

Figure 1: Mr. Latimer working with a third grader

first meeting was held at the beginning of December 2007.

Students were given very structured lessons for a majority of the year. The purpose of these lessons was to familiarize the students with Scratch and show them some of the many possible ways to use the program. In April, Mr. Allard and I decided that it was time for the kids to work on their own individual projects.

After spending most of April working on individual projects, Mr. Allard and I introduced the final project for the students: "Kitty Plays Football." This final project was a little more structured than simply letting the students work on their own, but still featured many different challenges that the children were encouraged to attempt. The final project filled up the rest of the year, ending with a little certificate ceremony for the students on June 5th, 2008.

3.2 Lessons

Each class lasted from 30-45 minutes, depending on the age of the students attending and the schedules that their teachers had set. As was mentioned before, much of the year was spent walking the students through very structured lessons in order to provide them with a basic understanding of Scratch. Each lesson began with the students signing in to the lab (for attendance purposes) and quietly sitting down at their computers. Once everyone was

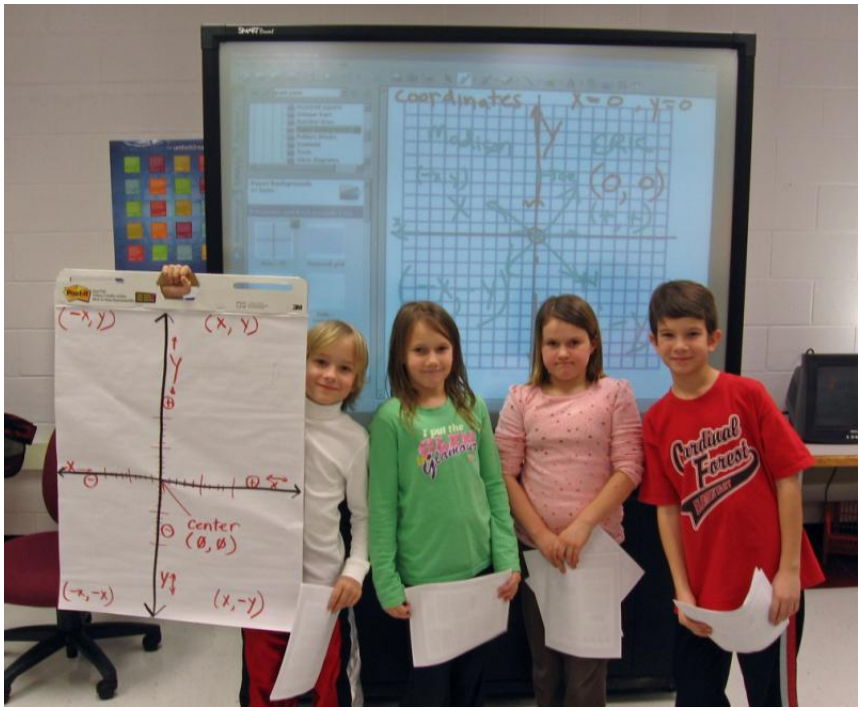

Figure 2: Students learning about the coordinate axes

seated, we began teaching. These teaching sessions generally consisted of Mr. Allard or I creating a small program with the kids which was based on the technique that was being taught that day. For example, our first programs were centered around telling a sprite to "move" to different places on the screen and using the "green flag" to start a script.

3.3 Topics

The actual number of topics that were covered was dictated by how quickly the students were able to move from one topic to the next. The number of topics that we did in fact discuss was rather extensive considering that the students had never seen the program before. Some examples of the Scratch related topics include: if-then statements, basic iteration (the "repeat" and "forever" loops), and custom sprite and stage creation. However, it was very important to Mr. Allard and I that the students learned about more than just computer science. We used Scratch to teach the students about other things such as the coordinate axes, negative numbers, degrees, and angles. Some of our younger students were learning these topics multiple grades before they needed to according to the SOL and POS benchmarks.

The potential for using Scratch (or computer science in general) as a

teaching tool is not limited to mathematics. For example, students can work on their language arts abilities by working with each other and presenting their projects to the class. Scratch program scripts are written in such a way that they can almost be read as a story, and this helps the kids connect one piece of code to the next. Some of our youngest students (kindergarten and 1st grade) were barely old enough to read the actual words, yet the color and simplicity of the blocks of code enabled them to actually do some programming. Scratch is a powerful way to bring many different subjects together that might normally be kept apart from each other in the classroom.

Mr. Allard and I were hardly able to teach the students everything there is to know about Scratch. One example of a topic that I feel like would have been good to touch upon is using different "control" blocks (i.e. pairing a "forever" loop with an if-then block) in combination with each other. Attention was given to most of the unique blocks individually, (for example one day the lesson might be on using the "repeat" block, while another day might be spent working on broadcasting) but there wasn't enough time to specifically teach the students to use these blocks in tandem with one another. Mr. Allard and I hoped that the students would be able to figure out this technique on their own. Scratch is a program that encourages individual exploration, and Mr. Allard and I wanted the students to take the initiative to do a little exploring of their own once we let them create their individual projects.

3.4 Individual Projects

Scratch participants were allowed to create their own individual projects during the month of April. All of the students eagerly anticipated this day when they could stop listening to Mr. Allard and I and start doing what they really wanted to do. In order to prevent absolute free reign, Mr. Allard created a rubric detailing some small requirements for each individual project. Each individual project was required to contain: one custom sprite, one custom stage, one sprite that "talked," and some sprite interaction. By the end of the month, the level of achievement was varied to say the least.

Sophistication of the students' programs understandably varied by grade level, although this variation was less distinct in the older (4th-6th) grades. Many of the students used mainly the most basic blocks (i.e. "move," "say," and "turn") over and over again with no regard for the basic loop techniques that we had discussed earlier in the year. This seemingly low retention rate

was attributed to the amount of freedom that we gave the students in creating their own projects.

The kids placed a large and unexpected amount of effort into creating their own custom sprites and stages. Mr. Allard and I feel that the amount of time spent doing this took away from the necessary time needed for any student to really think through his/her program. We wanted the students to spend more time programming than drawing. One possible reason that the students spent so much time creating sprites and stages as opposed to any actual programming might have been the paint-like program used to create them is something that the students were already familiar with. Programming for the students was a new and challenging experience and it seems like they weren't quite ready yet to handle much on their own. This unfortunate but enlightening result led me to create a final project with a little more structure.

3.5 "Kitty Plays Football"

I designed the final project to be structured in a way that Mr. Allard and I could continue to remind the kids of what needed to be accomplished, but I also tried to leave some room for creativity on the part of the kids. In order to avoid the sprite creation problem that plagued the last project, I went through the sprite creation process with the students so that everyone had similar sprites. After this, I walked the kids through the creation of the basic program. This basic shell had the kitty simply kicking the football, followed by the football going straight to the middle of the goalposts. After everyone had this basic shell, they were given a sheet with a number of challenges that they could attempt. Some of the tasks included making the football spin, making the football come back to kitty without dragging it, and adding a "score" variable.

The most difficult challenge for the students was making it so that kitty had the possibility of missing the goal (not all kickers are perfect). Accomplishing this task requires the use of the "pick random" number function, something that was not previously introduced to the kids. I included this challenge to see how many students were willing to experiment and use things that we had not necessarily taught them before. None of the students who attempted this task (without any help that is) got it 100% correct, but some did in fact get fairly close. A few students were able to make the football miss the goal sometimes, but they got stuck when they were faced with a

Figure 3: The Kitty Plays Football project logo

need to tell the program that the football missed the goalpost. However, the kids who tried to solve the problem demonstrated that they understood what the challenge meant. That was encouraging because it meant to me that they're starting to grasp the basic computer programming concept where the computer only knows as much as the programmer tells it.

3.6 Resources

The Cardinal Computer Lab features about 30 student computers along with a teacher workstation and another computer connected to a SMART Board. While a larger room would be appreciated so that more students could be accommodated at one time, these resources were sufficient for the project. It would also be difficult to handle any larger number of kids without more consistent parent help. There was one consistent parent volunteer (Mrs. Caroline Peck) during the year to help Mr. Allard and I manage the students, but there were a couple of other parents who came sporadically to work with the kids. Parent participation is a very important part of this program and it's important that they not only stay informed about the project, but participate in it as well. Mr. Allard was in attendance at every Scratch session, and I tried my best to attend every other one.

Figure 4: The Cardinal Forest Computer Lab

3.7 Subjectivity of Results

To put data into a chart or graph for this project would be difficult, unless something of an assessment was offered to the kids at one time or another. Mr. Allard and I have been reluctant to give such an assessment because we're afraid it would discourage some of the kids from participating in the program. Thus, proper (hopefully quantitative) assessment of student progress has become a goal for the next year of the program. The data generated from this experiment is rather subjectively based on my personal experience teaching and working with the elementary school students.

3.8 Future Research

There are many different things left to explore for next year when two new Thomas Jefferson students (Crystal Noel and Jessica Gorman) will continue the program. Mr. Allard will focus on finding alternative ways to assess the students, while also working on introducing new programming environments and opportunities. There is a possibility (depending on grant applications) of introducing Cricket next year, as well as Alice, Squeak, and/or Logo. The program will definitely continue next year and hopefully for many years to come.

4 Results and Discussion

Hopefully this project will encourage the implementation of a simple computer science curriculum in all elementary schools. The earlier that kids can start to program and become interested in computer science, the better. The computer has the potential to start a digital revolution in learning, not only in math and science but in English, social studies, and other subjects as well. By the end of this initiative's first year, the students have made obvious progress with respect to their familiarity with programming and Scratch. How young is too young to start teaching kids how to program? The time that I've spent with the students has shown me that you would be hard pressed to go young enough.

References

[1] Burd, Leo and Robbin Chapman. Beyond Access: A Comparison of Community Technology Initiatives. N.p.: n.p., 2002.

[2] Johnson, Carolyn Y. "With Simplified Code, Programming Becomes Child's Play." The Boston Globe 15 May 2007. 10 June 2008 <http://www.boston.com/news/education/k_12/articles/2007/05/1/with_simplified_code_programming_becomes_childs_play/>.

[3] Papert, Seymour. The Children's Machine: Rethinking School in the Age of the Computer. New York: Basic Books, 1993.

[4] - - -. Mindstorms: Children, Computers, and Powerful Ideas. New York, Basic Book, 1980.

[5] Resnick, Mitchel. "Rethinking Learning in the Digital Age." The Global Information Technology Report: Readiness for the Net-worked World. By G. Kirkman. Oxford, UK: Oxford University Press, 2002. 32-37.

[6] Sheehan, Robert. Children's Perception of Computer Programming as an Aid to Designing Programming Environments. Preston, UK: n.p., 2003.