REVIVING A LOST LANGUAGE

Why computer programming should be taught in schools again

Sandra Leiterman

Kansas State University

EDCI 750 Emerging Technologies

Dr. Clark Harris – Instructor

4/28/2014

The first personal computer, the Simon, was introduced in 1950 by Edmund C. Berkeley, President of E.C. Berkeley and Associates. Simon was just a basic computer that could do simple algorithms similar to a complex calculator. Within 25 years, people began to see just how cool a personal computer could be. In the mid 1970's there were two companies competing in the personal computer market: IBM and Apple. Their models were more complex than the Simon, but both computers were still very basic computers compared to what is available today. In 1984, 8.2% of US homes had a computer, by 2012 the number increased to 78.9% (US Census Bureau, 2014). Along with the personal computer, programming languages actually existed well before BASIC programming language of 1964. However, the introduction of BASIC programming brought computers to a whole new level and today, there are over 2500 documented programming languages, though many are no longer used (Kinnersly, 2014).

Today, computers are a very integral part of K-12 education. There are computers in nearly every classroom across America, and the new Common Core State Standards have technology standards embedded into them. The use of technology will further increase as we move forward in time. Teachers and students are learning more digital technology with the use of Web 2.0. Web 2.0 is not a *new* internet, but rather a new style of websites designed with collaboration and interaction in mind. Social media such as FaceBook and Instagram, and blogging sites such as WordPress and Tumblr, are all websites designed for Web 2.0. Websites such as these are much more complex than the websites and web tools available just 10 years ago. The technology and consumer demand changes daily; therefore, the companies behind the websites, web tools, smart phone apps and software must employ some of the smartest, most

dedicated people in the workforce. They must employ a great number of people known as programmers, software engineers, or the buzzword of the year, "coders."

Coders are the people who write the programming code to make the apps, web tools, websites, software and video games work. While there are over 2500 programming languages, most programs are written in one of ten popular programming languages. Two programming web based programming languages for beginners have emerged since 1999, one is Alice and the other is Scratch. Scratch is a programming language and online community developed by MIT, where students can create and share digital stories, games, and animations. Alice is a 3D programming platform, developed by Carnegie Mellon for making digital stories, interactive games or videos to share on the web. Both programs were developed for use by children as young as 8 years old, and feature a drag and drop programming style making it easy for children to learn the basics of programming, without typing lines and lines of code.

Once Alice and Scratch are mastered, it's time to move on to more advanced programming. According to Rich Hein, the top programming languages most in demand by employers are:

SQL- Structured Query Language is a programming language in which was designed for editing and querying data in relational database management systems. An example of a website designed using and SQL database would be a shopping website such as dillards.com or shoebuy.com. Both sites allow users to search the database of product offerings, and narrow down the results using predetermined filters.

Java and JavaScript – both are similar programming languages in that they are used in interactive web applications. Java is a more technical language and less forgiving than the newer JavaScript. Java has more programming options, but must be translated to smaller language pieces before it can be used on a computer; once the program is compiled, it cannot be edited. JavaScript on the other hand, as fewer option, but can be ran "as-is" and also has the ability to edit at any time.

HTML- language is the building block of nearly all websites. Used alone, a user will get a standard page by page website. However, HTML can be combined with other programs such as JavaScript to create a flashier website with more appeal to the end user.

C - A basic and straightforward programming language developed in early 1970. A coder must tell the program to do every step individually. All the individual steps can make a program very lengthy and susceptible to syntax errors.

C++ and C# - both programs are derivatives of C, and essentially allow the coder to compact lines of code into "objects". This makes for less lines of code. C++ was introduced in 1983, and C# in 2000 for specific use with Microsoft's .net platform.

Python, Ruby, Perl - programming languages for interactive web tools, games and apps.

Many of the websites, apps and web tools used in the classroom are written with one of the above programming languages. While students are doing more of their learning digitally, what the students aren't learning; is how to read and write programming language. In the early 80's, many high schools taught computer science, more specifically BASIC, Logo and Pascal programming (Kafai and Burke 2014). Today, there are nearly 42,000 high schools in the United

States, and only about 2100 of them offer AP Computer Science (College Board, 2012). The number of classes offered has declined by 17% since 2005. However, the need for programmers is increasingly real. The US Labor and Statistics Board estimates there are currently just over 340,000 jobs in the programming field, and the number will grow to nearly 1.4 million by the year 2020 (2012). As educators, we strive to make our students college and career ready, yet American educational institutes overlook one of the highest need career fields. We must figure out a way to bring programming courses back into schools nationwide.

To first learn the basics of computer programming, one must first be taught computational thinking. According to ISTE, "Computational thinking (CT) is a problem-solving process that includes creativity, innovation, logic, reasoning, and analyzing" (2011). These are all 21st century skills which will benefit students on all levels of their academic careers from middle school to high school. Creativity and innovation allow students to be active participants in their own learning. Science and math use logic and reasoning in showing relationships of parts to whole. A student who learns the foundations of computational thinking will understand how to use digital tools to solve future problems, and will also most likely attain higher achievement in all other content areas (2012).

Integrating computer programming into a classroom lesson plan is not the easiest approach to the inclusion of coding in schools, but with a little time and effort, in can be done. Many web tools and apps make it easy to integrate computer use in schools, but coding can be a more difficult task. Some shy away from it simply because they are intimidated by the complexity, or feel they don't have the proper certifications. If you can have students work within the limitations of an already established program, wouldn't it be far more fascinating for

them to design their own web tool to solve the problem? Coding doesn't have to wait for AP Computer Science in high school; students as young as 4th grade are having success with beginner programs such as Alice and Scratch. Janice Mak, a teacher in Phoenix, AZ began teaching coding to her 4th grade students in hopes of leading her "students to new ways of thinking, understanding and communicating in a digital world" (2014). Her first lesson was about idioms, where she included the task for students to create an animation in Scratch illustrating idioms. Mak then asked her students to reflect on their experience, and the following is the response from one fourth grade student:

> I am working on an idiom project —a taste of your own medicine. It was pretty challenging because most of the sprites they had were not what I needed so I had to create my own and my own background and fix things up and reprogram it to make it better. I had to make my mad scientist disappear. When I was trying to do that I tried to make my own block and typed in "disappear," but when I clicked on that for him, he just stayed there. Then I had to make him move 240 steps so you didn't see him anymore, but you could still see his hand. I'm trying to make him "disappear," but it's hard because I tried to get the background to go over him, but that didn't work either (so)… I'm looking around Scratch to see how I can program it to set itself up all over again (2014).

Reflective learning is a 21st century skill, as well as communication, collaboration, and digital literacy- all of which are accomplished by coding an animation in Scratch. For more information on how Janice Mak includes coding in her everyday classroom, visit her blog at http://supercodingpower.blogspot.com/.

Inquiry based learning is another catch phrase of the year. Teachers at Centennial Middle School in North Boulder, Colorado believe not all inquiry projects are created equal. They sought to tap into their students' interests to create the best project possible. They incorporated the popular video game past time of their students, and took it a few steps further. While playing video games requires complex knowledge and problem solving skills, creating those games involves situated cognition, which is where learners develop certain skills from within a group. "Research has shown that students who are exposed to situated cognition methods, which involve immediate real-world application and allows players to determine their own objectives, see greater value in learning" (Holbrook, Savignano, & Williams, 2014). The cross curricular project included designing an app, creating a company, and making a video to promote the app and the company. Each of the tasks included standards from both math and English CCSS, as well as inquiry standards of science, and societal needs of social studies. At the conclusion of their first round of the unit, they had students attempt to publish their apps- two of which were successful and are available in the Apple App store today.

Coding in the classroom may not be feasible if a school district doesn't have the computer resources available for coding. However, there are several ways students are learning coding basics such as logic, sequence, algorithms, binary numbers and data compression without ever logging on to a computer. Christian Thompson is a teacher in Tokyo, Japan who teaches ICT to middle school students. When questioned on why he teaches programming when we have so many web tools and apps so readily available, he responds:

> Even though we have calculators, we still ask students to learn how to add, subtract, multiply, and divide. The principle is the same. The process and basic knowledge are in and of themselves useful and the skills gained are transferable to other domains (2011).

On his blog site, Thompson incorporates many ideas on how to teach coding without a computer, which is one of the biggest obstacles to teaching coding in schools today. Thompson provides a paper and pencil version of basic object programming and presents different challenges to his students, including simple task completion, and even accomplishing the task in the fewest steps possible. His activities and lessons are available for free download at: http://christianthompson.com/node/31. Even more resources and classroom ideas can be found at http://csunplugged.org/.

In the past few years, through initiatives such as "an hour of code" and "why computer science", an increasing number of schools in America are figuring out ways to incorporate coding into their curriculums. Some schools are able to create an entire class dedicated to coding, while others have implemented coding through extracurricular activities taking place at times such as lunch, study hall or after school. Whichever method is chosen, there are many free resources available for teaching coding in elementary, middle and high schools.

Code.Org is one of the most comprehensive websites launched with a mission to bring coding into public schools, easily and FREE! It all began with an "hour of code" campaign coinciding the national computer science week in 2012. It has since grown and boasts that over 33 million people have tried and hour of code. Through funding from various foundations and corporations, code.org is able to partner with schools and districts across the nation to bring the lost language of computer programming back in to our schools. Currently, code.org is offering grants to teachers to incorporate coding at their schools. They also offer district training and PD stipends for those who agree to provide coding throughout the entire school district.

Scratch is a free programming and collaboration site developed my MIT and credits its continued success through grants and corporate sponsorships. Programming in Scratch allows students to create animations of varying lengths, and share their finished product with an online community. As teachers expand digital storytelling in their classrooms, Scratch allows the programmer to write the code to create the animation that could easily be used as a form of digital storytelling. Currently, Scratch has over five million animations shared on their site, and they have now expanded their programming efforts to include children as young as five. Tufts University, through a grant from the National Science Foundation and permission from MIT to use the Scratch platform, developed an age appropriate interface that will allow children ages 5-7 to drag and drop commands to create simple animations. Their Kickstarter campaign exceeded the initial fundraising goal and will allow the release of free apps for IPad and Android in summer 2014. Scratch will also be developing resources and curriculum for parents and teachers.

For more advanced programming needs, codeacademy.com has lessons in HTML, CSS, JavaScript, Python and Ruby and more. Through prebuilt lessons and activities, a learner completes each lesson and gains experience points. Once enough experience points are gained, access is given to create a custom curriculum to use in nearly every secondary educational setting.

The need for programming is real. Learning to code for software, web tools and apps builds collaboration and communication skills, and instills critical thinking and problem solving strategies not otherwise taught in an average classroom. Research concluded by Justin Solomon and Peter Rusev indicate that early acquisition of computer programming is far more beneficial in developing 21st century skills than waiting until high school (2008). Developing apps,

software and web tools require teamwork, algorithmic thought processes, and problem solving skills. Technology apps are constantly being updated for fear of becoming obsolete, and the future of technology apps and the internet dependson coders. Currently, only 2% of the STEM graduates have a computer science degree, while 40% of the available jobs are in the computer science field (code.org 2013). The College Board reports that only 3,000 of the nearly 3.6 million AP tests were in computer science (2012). The gap between graduates in the programming field and jobs that need to be filled will only continue to grow if efforts aren't made early enough in a student's academic career. Computer programming can no longer wait for AP Computer Science in high school, it must start earlier. Recent research from ACT shows that "the level of academic achievement that students attain by eighth grade has a larger impact on their college and career readiness by the time they graduate from high school than anything that happens academically in high school" (2008). But it's more than just filling the jobs of the future. A student fluent in any programming language will have far more problem solving skills than one with no fluency. Computer science is a foundation science. The ability to create a program far exceeds the ability to use a program. If students are taught how to read and write a paragraph, a magazine or a book- all of which they will use throughout their lifetime- shouldn't they also learn to read and write the code that creates the apps or the web tools they will also use over a lifetime? Starting a programming class or club is easy. It only needs to start with "an hour of code". When will you start your hour of code?

Works Cited

Bidwell, Allie. "Hour of Code Initiative Engages Middle School Girls in Computer Science." *US News and World Report* 10 Dec. 2013: n. pag. *www.usnews.com*. Web. 24 Apr. 2013.

"Computational Thinking Toolkit." *Computational Thinking Toolkit*. N.p., n.d. Web. 25 Apr. 2014. <https://www.iste.org/learn/computational-thinking/ct-toolkit>.

"Computer and Internet Use." - *Computer and Internet Access in the United States: 2012*. N.p., n.d. Web. 27 Apr. 2014. <https://www.census.gov/hhes/computer/publications/2012.html>.

"Create stories, games, and animations." *Scratch*. MIT, n.d. Web. 26 Apr. 2014. <http://scratch.mit.edu/>.

Freedman, Terry. "Coding is not debugging." *Tech and Learning* 12 Apr. 2014: n. pag. *www.techlearning.com*. Web. 24 Apr. 2014.

Hein, Rich. "10 programming languages that are in demand by employers." *Computerworld*. Computer World Magazine, 11 June 2013. Web. 25 Apr. 2014. <http://www.computerworld.com/slideshow/detail/98085#slide1>.

Kafai, Yasmin B., and Quinn Burke. "Computer Programming Goes Back to School: Learning Programming Introduces Students to Solving Problems, Designing Applications, and Making Connections Online." *Education Week* 1 Sept. 2013: n. pag. *www.edweek.org*. Web. 23 Apr. 2014.

Kinnersly, Bill. "The Language List." *The Language List*. N.p., n.d. Web. 24 Apr. 2014. <http://people.ku.edu/~nkinners/LangList/Extras/langlist.htm>.

Mak, Janice. "More Scratching." *Super Coding Power*. N.p., n.d. Web. 27 Apr. 2014. <http://supercodingpower.blogspot.com/2013/10/more-scratching.html>.

Moyle, Kathryn . "Building Innovation: Learning with technologies." *Australian*

Education Review 56 (2010): 24-27. *www.acer.edu.au*. Web. 25 Apr. 2014.

Savignano, Mark, Mia Williams, and John Holbrook. "LEARNING CONNECTIONS: Yes, Your Students Can Create Games That Land in the Apple App Store!." *ISTE Learning and Leading*. ISTE, 27 Jan. 2014. Web. 26 Apr. 2014. <http://www.iste.org/learn/publications/learning-leading/issues/l-l-february-2014/learning-connections-yes-your-students-can-create-games-that-land-in-the-apple-app-store!>.

"ScratchJr | Coding for Young Kids." *scratchjr*. N.p., n.d. Web. 26 Apr. 2014. <http://scratchjr.org/>.

Solomon, Justin , and Peter Rusev. "Early Acquisition of Computer Science." *Early Acquisition of Computer Science*. Stanford University Computer Science, n.d. Web. 25 Apr. 2014. <http://cs.stanford.edu/people/eroberts/cs201/projects/early-acquisition-of-cs/index.html>.

"Summary." *U.S. Bureau of Labor Statistics*. U.S. Bureau of Labor Statistics, 28 Jan. 2014. Web. 26 Apr. 2014. <http://www.bls.gov/ooh/computer-and-information-technology/computer-programmers.htm>.

"What are 21st Century Skills?." *What are 21st Century Skills?*. N.p., n.d. Web. 26 Apr. 2014. <http://atc21s.org>.