EARLY COMPUTER SCIENCE EDUCATION

Cristopher Laney

Oregon Institute of Technology

Early Computer Science Education

Abstract

There are many benefits to teaching Computer Science skills early on in a child's academic development. Programming can teach children how to approach problems in a more efficient and structured way which will help them in classes across all subjects. Computer Science education can seem intimidating at first glance, but after looking at the tools available it is apparent that teaching children how to program can be easy and fun for both the students and the teachers. Little previous knowledge is needed to start teaching these skills to children. Languages like Scratch have a massive user base and enough documentation to quickly get any teacher or student up to speed.

By Cris Laney

Technical Report Writing

6/1/15

Contents

List of Figures

Introduction

Computers have been dominating the workplace since the early 2000's. If used properly they can make our lives much easier in almost every aspect. Computer Science is a subject of increasing popularity, and teaching it to children from a younger age will help them develop critical thinking skills, and teach them how to use computers efficiently. These skills will benefit them later on regardless of their career path. Programming, an important part of Computer Science education, can teach critical thinking and basic computer skills. Programming can be taught in a variety of ways, and with so many tools available, it can seem overwhelming at first, but choosing the right tool is actually fairly simple.

Computer science educational tools range from simple animations, to tangible interactive surfaces, and even robotics. The goal of these tools is to make programming accessible while allowing enough room for future advancement.

The easiest way to teach programming is to use one of the many tools already available. Allen Tucker, who wrote about potential curriculum changes for grades K-8, recommends that students should be provided with basic computer science concepts that help them think like a programmer [1]. There are several tools that are perfect for teaching the "algorithmic thinking" that Tucker recommends. In another paper, *Programming Without a Computer: A New Interface for Children under Eight*, the authors state the following design goals:

> "1. activities are open-ended and discovery-oriented, allowing children to be actively involved in the learning process;
> 2. interaction encourages child-initiated play;
> 3. experiences involve active manipulation and transformation of real materials; 4. entry level knowledge and experience is kept to a minimum;
> 5. provision is made for children's varied skill and ability levels;
> 6. construction activities that involve design, creation and evaluation processes form the basis of interactions." [2]

These design goals are important to keep in mind both when developing and searching for tools for children in the K-8 age group. Some of these tools are free, and some can cost upwards of 700$ for assorted hardware.

There is an important distinction between programming and computer science. Programming is only one aspect of Computer Science. Computer Science includes several subjects of study including program complexity and various mathematical disciplines, but programming is at its core. Programming can also teach life skills extending beyond the world of computer science.

Benefits of Teaching Children How to Program

Trend Toward Computer Based Jobs

Today's job market shows a trend toward computer based jobs, and even jobs that are not computer based usually require some level of interaction with computers. Teaching children from a young age how these computers work could massively benefit them and prepare them for the workforce. There are currently not enough properly trained STEM majors in the job market, and teaching children the fundamentals early on could assist in alleviating the STEM worker shortage. For example, in a study conducted by Brookings where STEM related job postings were monitored for an extended period of time, the researchers found, "Almost half of the 239,000 jobs posted for at least 70 days require high levels of STEM knowledge." And that, "One-fifth of vacancies are posted for at least 70 days." [3] This study shows that STEM majors are desperately needed in the job market, and the shortage does not show any signs of slowing.

Increasing Problem Solving Skills

Learning how to program gives students important critical thinking skills that extend beyond Computer Science. Computer Science teaches students to take large problems and break them up into smaller and smaller pieces. These skills are useful in subjects like math and science, but can be applied to any large project. The best way to teach this divide and conquer approach is by teaching children how to program. Programming languages are created in such a way that the problem must be broken down into its smallest parts before it can come together as a whole. The main differentiator between writing a program and solving a large equation is that when programming it is often literally impossible to solve a problem without breaking it into manageable pieces first. In subjects such as math, a talented student can often do large portions of the problem at a time without going through each individual step.

Breaking up a problem in this way is sometimes referred to as creating "subgoals." It is apparent that students often approach a problem incorrectly when they are first starting out, so teaching children how to ask the right questions before they encounter more advanced problems would be beneficial. For example, the article *Employing subgoals in computer programming education* made the observation that, "novice physics students tend to focus on the contextual features of problems (whether an example used a ramp) instead of the structural features (which law applies to the problem)" [4]. By teaching these skills via programming instruction early on, educators can prevent students from making these same mistakes in the future. Even though some subjects, such as physics, can teach many of the same skills, those skills may be obfuscated by the subject itself. In the subject of Physics the reward lies more in the answer rather than the process, whereas when programming the reward lies in the process of creating the program.

Computer Science teaches the subgoal process in a more efficient way than subjects such as Math or Physics. According to an article from Kent State University, where children were given different tests before and after being given programming lessons, students who learned some basic programming concepts before made far fewer errors on the post-test as opposed to the pre-test [5]. This article is evidence that programming is an effective way of creating a solid foundation for problem solving skills.

Best Ways to Teach Children How to Program

There are several tools available to teachers and parents who wish to teach children how to program. This section is an evaluation of several of the tools currently available. Some of these tools have failed to make it into the mainstream, but had important design goals to keep in mind. Scratch has increased in popularity and has a thriving online community. Tur Tan and Toque, however have fallen by the way-side, due in part to their high price points and their lack of practicality. Tur Tan and Toque did end up making important contributions to future languages that incorporated tangible and graphical interfaces.

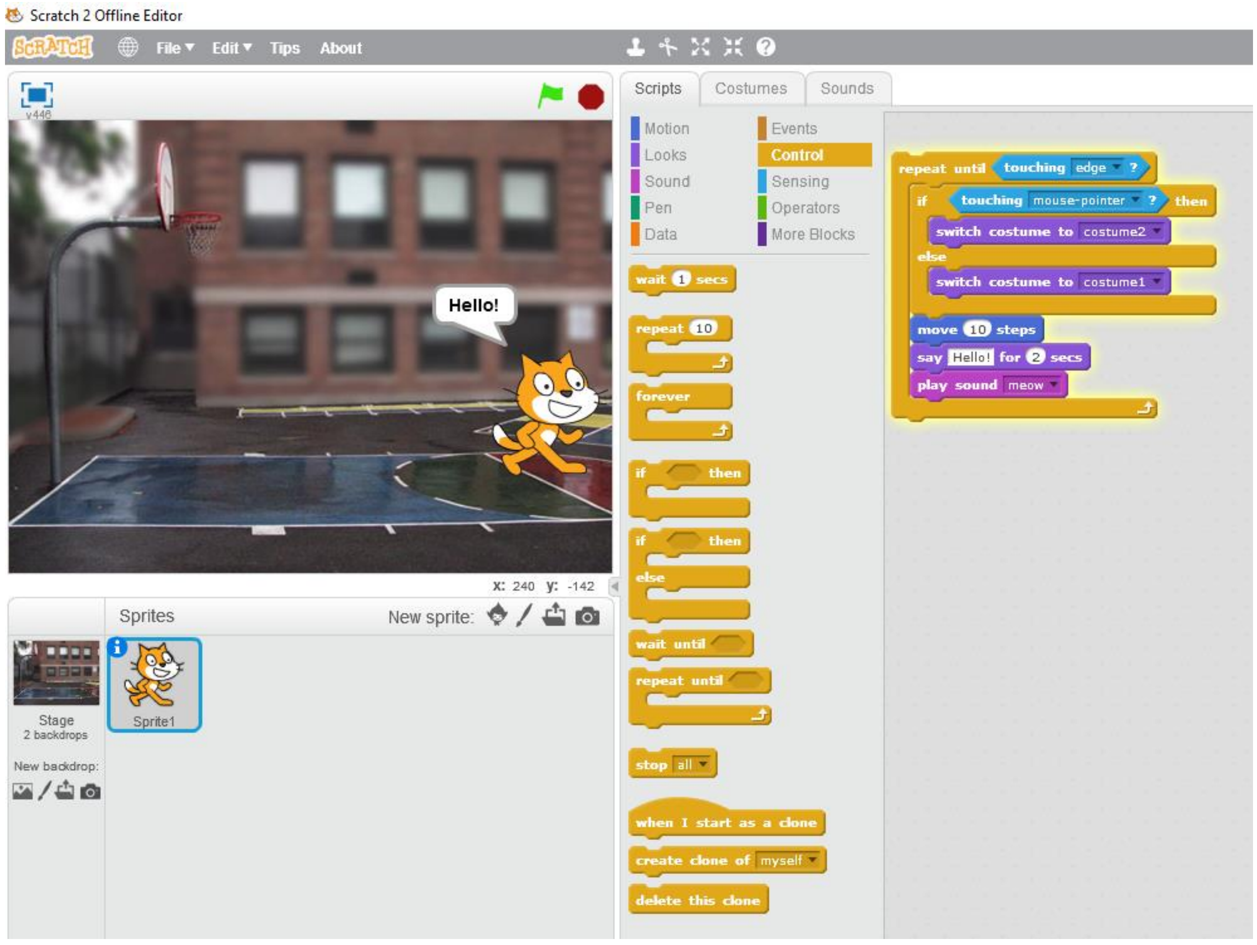
Tools Available for Computer Science Education

The choice of tool is important when it comes to teaching children how to program. Most of the tools available are programming languages with a high level of abstraction. Even though these languages have an impractically high level of abstraction, they teach the same concepts that are fundamental to Computer Science education. An important aspect of these languages is that they provide instant gratification to students usually in the form of audio visual feedback. Every program the students write, results in an instantly viewable animation. Instant gratification is important for young students in order to make the exercises enjoyable. There is an added benefit to making the languages easy to pick up, in that it also makes it far easier for teachers to learn and understand. Since it is not a “true” language, even if the teacher does not understand all of the intricacies of programming and computer science they can still teach it to the class without teaching them poor programming habits.

There seem to be two ways to interact with a program using these types of tools. The first is through a Graphical User Interface (GUI), and the second through a Tangible User Interface (TUI). The Graphical User Interface allows students to manipulate programs through icons and pictures on screen. The Tangible User Interface allows the student to manipulate programs through physical objects like blocks with different actions drawn on them. Some argue that Tangible User Interfaces are better for young children, because they are more similar to real world interaction [6]. The major downside to these “Tangibles” is that the hardware involved can exceed the budget of many schools and families.

Scratch

Scratch is a programming language developed by MIT for young children. It allows students to create animations and characters based on premade programming functions, commands, and conditions.



```
repeat until touching edge ?
    if touching mouse-pointer ? then
        switch costume to costume2
    else
        switch costume to costume1
    move 10 steps
    say Hello! for 2 secs
    play sound meow
```

Figure 1. A simple Scratch Program

As shown in the figure above, all the student needs to do is drag and drop a command into a box on the right hand side, and the sprite on the screen will execute the commands listed. Scratch was intended for teaching basic programming concepts to people who had never programmed before [7]. The developers of Scratch did research about the best ways to teach young people how to program. In their research they found that, “programming languages should have a ‘low floor’(easy to get started), and a ‘high ceiling’(opportunities to create increasingly complex projects over time).”[7] These concepts are important, especially at a young age, because the students may be incapable of processing or understanding the long term benefits of what they are doing. It provides a new and exciting entry into the world of Computer Science.

Another goal of Scratch was to make the language, and development more social than conventional programming languages [7]. They created a web-site where people could post and discuss their projects. Although this sounds like a good idea, it may discourage parents from downloading Scratch for their children. In communities where computers, social networking, and programming are not common this online community could be a scary idea given the reputation internet communities have in the media. Parents may be afraid to allow their child to participate in these forums and somehow associate the forums with the completely separate programming language. Unfortunately, the places where these fears exist may be the places where Computer Science Education is most needed.

Tur Tan

TurTan is described as being a “tangible language” by its creators. It allows users to create characters, draw paths, and program movement. TurTan was created with the same ideas as Scratch in that it is intended for non-programmers and children to understand how programming works [8].

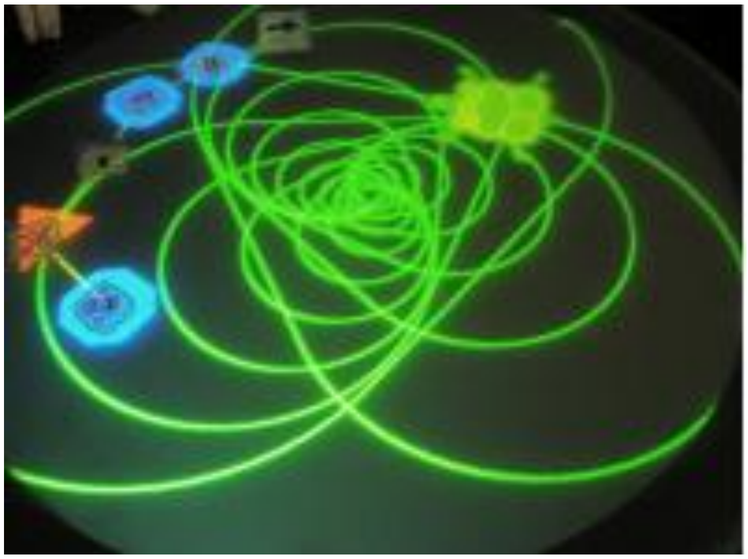


Figure 2. Tur Tan Tabletop.
D. Gallardo, C. Julia and S. Jord ` a`, "TurTan: a Tangible Programming Language for Creative Exploration", Music Technology Group, Universitat Pompeu Fabra, 2016.

Using the graphical tabletop shown in Figure 2, TurTan allows users to directly interact with tangibles which are images on the tabletop that represent different programming concepts and instructions. The benefit of using tangibles is that students do not need to worry about the intracacies of syntax. These abstractions can be beneficial especially to young children. It may help avoid frustration, and encourage them to delve deeper into their projects. Even though TurTan has a high level of abstraction, it still teaches children how to break down problems. Students will have to break down the path they want the turtle to follow in order to get it to function as expected.

There are many issues with using TurTan as a tool to teach children how to program. For one, because the entire system is built into a tabletop, it can only serve one purpose. It is also very large and implements touch sensitivity which could cause the device as a whole to be expensive. TurTan is still in development, so though it shows promise, cannot currently be deployed on a large scale.

Toque

Toque is another tangible programming language, though not on the same scale as TurTan. Toque is a programming language that uses recipes and cooking instructions to teach children the basics of programming. The Wii remote and Nunchuck are used to create gestures that the Toque programming language will recognize as certain commands. Representing a program as a recipe, as shown in Figure 3, allows students to understand programming through a familiar format. Instead of simply teaching the students how to program, it is represented in an entertaining and understandable way which is important when teaching students at the elementary school level. The incorporation of gestures may also aid the students by providing even more abstraction without making the device it runs on overly expensive or impractical. It is emphasized in, *Toque: Designing a Cooking-Based Programming Language For and With Children* that the researchers continued improving on the Toque prototype after getting feedback from the children that used it [9]. Creating multiple iterations is important when writing any program, but it is especially useful when creating software for an audience that is so far removed from the developer. These students may have never even used a computer before, so the language needs to be tested and modified based on the experiences of the children who use it.

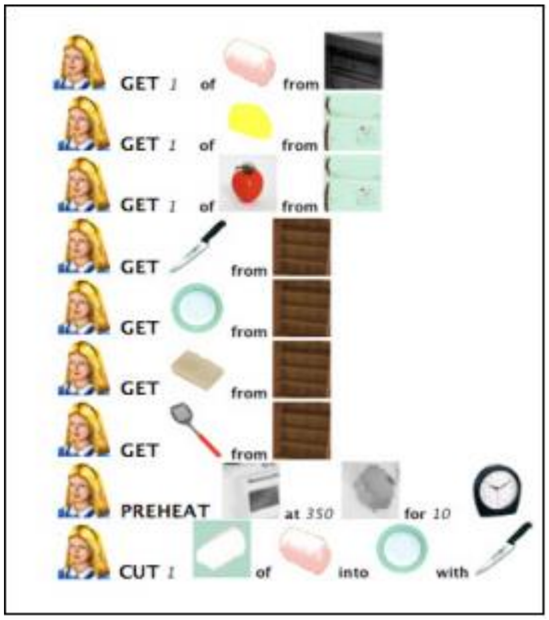


Figure 3. An example of a Toque program.
D. Gallardo, C. Julia and S. Jord `a`, "TurTan: a Tangible Programming Language for Creative Exploration", Music Technology Group, Universitat Pompeu Fabra, 2016.

Unfortunately, Toque has not gained much traction, but it serves as a great example of what introductory programming should look like. The language was tested several times and improved upon not only based on observations from the creators, but the experiences of the children that used it. These ideas and techniques are important to keep in mind when developing for children, and the Toque project should serve as an example of everything that developing for children should be.

Robotics

Robotics is a method of teaching computer science that has increased in popularity in recent years with the decrease of prices in both robotic parts and computers. Robotics is an effective way to teach young children how to program because, like the languages mentioned above, it gives a sense of instant gratification. The major difference between robotics and the previously mentioned languages, is that the robots built by students are physical representations of their code. These robots can interact with the world around them, and give students a better understanding of what their code is doing. Robotics is also a good example of the practicality of programming. Instead of creating a simple animation, students will see that their code can actually be used in a practical manner.

Lego WeDo Robotics and Creative Hybrid Environment for Robotic Programming

The Creative Hybrid Environment for Robotic Programming (or CHERP) allows students to use either a GUI or a TUI in the form of blocks as opposed to digital pictures. In the article *I want my robot to look for food*, it is stated that young children have a much harder time conceptualizing, or picturing programming concepts mentally [10]. Conceptualizing is a necessary skill when programming, so to make it easier the designers of these languages and robotics kits allow the tangible method as an alternative to the GUI. This study found that students in general preferred using the Graphical User Interface as opposed to the tangible one, but also showed that the group of children who were

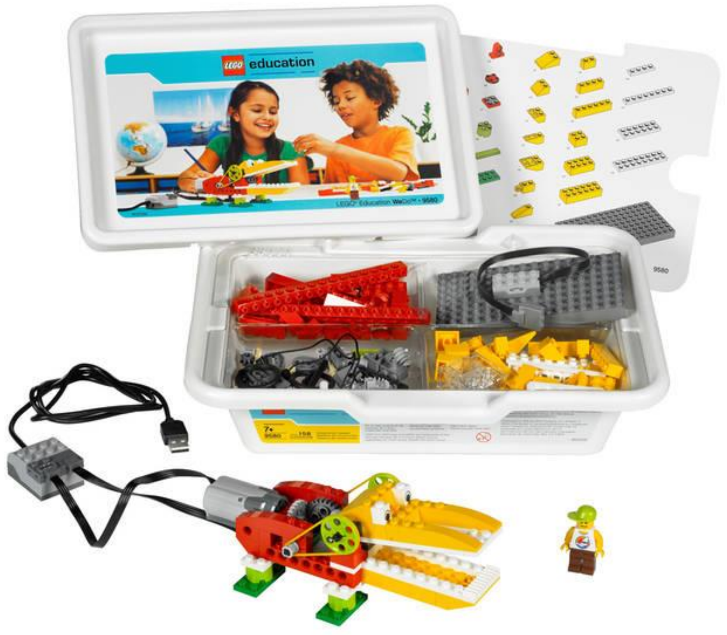


Figure 4. Photo of a Lego WeDo kit. http://education.lego.com/en-us/products/lego-education-wedo-construction-set/9580

introduced to the Tangible Interface first performed better when given different tasks [10]. The Lego WeDo Robotics kit pictured in Figure 4, is a great way to teach robotics, but it is a bit expensive. A package from Lego's website containing 8 kits costs approximately 750$. If a school is planning on improving upon an existing computer science program or they have a higher budget, the Lego WeDo kit may be the way to go.

PROTEAS

PROTEAS is a robotics programming language, intended for children. PROTEAS was used in a study in the journal *Personal and Ubiquitous Computing*, because the tools necessary for the PROTEAS Kit are cheap, portable, and durable relative to other TUI tools available for children. PROTEAS utilizes physical blocks that have representative pictures of the actions available to the Lego NXT robot. This block system allows children to physically interact with the program flow and better conceptualize what programming is about. The study also allowed students to use the blocks or a GUI to program the robot in order to accomplish certain tasks. The GUI has all of the same commands and actions as the block, but is entirely on a computer instead of on physical blocks.

According to *Evaluating children performance with graphical and tangible robot programming tools*, younger children accomplished the tasks in a shorter amount of time when using the blocks as opposed to the GUI. However, when the children in the 11-12 year-old group completed their task the results were almost the same [11]. This study proves that using a tangible interface can vastly improve student's ability to accomplish tasks with code.

Conclusion

The benefits of teaching children computer science concepts through programming is necessary in the increasingly technology-based world we live in. Computer Science concepts not only teach important computer-based skills, but also allow students to discover a new way to think about and approach problems. The tools available to teach entry-level programming are plentiful each offering advantages and disadvantages, however there is a tool for pretty much any price point or experience level. These tools can abstract away many of the details that make programming so intimidating, frustrating, and disheartening for many young students, and can prevent them from being permanently dissuaded from the vast technologies that surround them every day. Programming can teach them to embrace and use technology in a beneficial way.

Recommendation

There are two clear options when selecting a tool to introduce programming to children. The Lego WeDo kit, is the most effective way to teach programming concepts to students at the elementary level. Tangibles help students conceptualize programming in a physical way and can help them understand more advanced programming concepts in the future. The down-side to the LegoWeDo kit is that it may be too expensive for some schools or homes, and there is not as much support for it online as opposed to Scratch which has a very active online community. The benefit of Scratch is that it is completely free. All you need is a computer to run the Scratch software on. Scratch also has a massive and an online community where users frequently post their own projects online. An active community makes it easier for students and especially teachers to find ideas for projects, or seek out help when necessary.

Method	*Computer*	*Total Cost*	*Other Externals*
Scratch	Required	0$	N/A
Lego WeDo	Required	750$ (8 kits)	Lego Kit(Required), Tablet(Optional), Blocks(Optional)

Figure 5. Table displaying costs of Scratch and Lego WeDo

For schools that already have an established computer science, or programming curriculum and are seeking a more advanced route, the Lego WeDo kit is the best choice. It is slightly more advanced than Scratch, and provides students with more tangible results which provides a better foundation when children first start programming. However, if this is the first Computer Science class or program at a school that has no experience with programming, or the intention is for a student to learn on their own time outside of school Scratch is the best option. Scratch is perfect for these cases, because it is cheaper and has a much larger community, making it much easier to produce visible results in cases where there is not a firm support system in place.

References

[1] A. Tucker, "A New K-12 Computer Science Curriculum", International Society for Technology inEducation, 2004.

[2] P. Wyeth and H. Purchase, "Programming without a computer: a new interface for children under eight", *Proceedings First Australasian User Interface Conference. AUIC 2000 (Cat. No.PR00515)*.

[3] J. Rothwell, "Still Searching: Job Vacancies and STEM Skills", *Brookings*, 2014.

[4] L. Margulieux, R. Catrambone and M. Guzdial, "Employing subgoals in computer programming education", *Computer Science Education*, vol. 26, no. 1, pp. 44-67, 2016.

[5] D. Clements and D. Gullo, "Effects of computer programming on young children's cognition.", *Journal of Educational Psychology*, vol. 76, no. 6, pp. 1051-1058, 1984.

[6] M. Horn and R. Jacob, "Designing tangible programming languages for classroom use", *Proceedings of the 1st international conference on Tangible and embedded interaction - TEI '07*, 2007.

[7] M. Resnick, B. Silverman, Y. Kafai, J. Maloney, A. Monroy-Hernández, N. Rusk, E. Eastmond, K. Brennan, A. Millner, E. Rosenbaum and J. Silver, "Scratch", *Communications of the ACM*, vol. 52, no. 11, p. 60, 2009.

[8] D. Gallardo, C. Julia and S. Jord ` a`, "TurTan: a Tangible Programming Language for Creative Exploration", *Music Technology Group, Universitat Pompeu Fabra*, 2016.

[9] S. Tarkan, V. Sazawal, A. Druin, E. Golub, E. Bonsignor, G. Walsh and Z. Atrash, "Toque: Designing a Cooking-Based Programming Language For and With Children", in *CHI 2010: Cooking, Classrooms, and Craft*, Atlanta, GA, 2010.

[10] A. Strawhacker and M. Bers, "“I want my robot to look for food”: Comparing Kindergartner’s programming comprehension using tangible, graphic, and hybrid user interfaces", *International Journal of Technology and Design Education*, vol. 25, no. 3, pp. 293-319, 2014.

[11] T. Sapounidis, S. Demetriadis and I. Stamelos, "Evaluating children performance with graphical and tangible robot programming tools", *Pers Ubiquit Comput*, vol. 19, no. 1, pp. 225-237, 2014.