EUROPEAN JOURNAL OF STEM EDUCATION

European Journal of STEM Education, 2018, 3(3), 08
ISSN: 2468-4368

lectito

Coding and Computational Thinking in Early Childhood: The Impact of ScratchJr in Europe

Marina Umaschi Bers [1]*

[1] *Tufts University,* 105 College Avenue, 02155 Medford, USA

***Corresponding Author:** marina.bers@tufts.edu

Citation: Bers, M. U. (2018). Coding and Computational Thinking in Early Childhood: The Impact of ScratchJr in Europe. *European Journal of STEM Education*, 3(3), 08. https://doi.org/10.20897/ejsteme/3868

Published: September 6, 2018

ABSTRACT

In recent years, there has been an increased effort to introduce coding and computational thinking in early childhood education. In accordance with the international trend, programming has become an increasingly growing focus in European education. With over 9.5 million iOS downloads, ScratchJr is the most popular freely available introductory programming language for young children (ages 5-7). This paper provides an overview of ScratchJr, and the powerful ideas from computer science it is designed to teach. In addition, data analytics are presented to show trends of usage in Europe and and how it compares to the rest of the world. Data reveals that countries with robust computer science initiatives such as the UK and the Nordic countries have high usage of ScratchJr.

Keywords: early childhood, programming, computational thinking

INTRODUCTION

The idea of introducing computer programming in the classroom dates back from the late 60s'. At that time, Seymour Papert and colleagues at MIT developed the first programming language designed for children, LOGO. Using a simple textual language, children could type commands for a "turtle", so she could move around and draw geometrical shapes by dragging a pen (Papert, 1980). LOGO's popularity grew all over the world and new versions of LOGO were implemented in more than a dozen spoken languages on a variety of machines (The Logo Foundation, 2015).

However, although LOGO pioneered the growing trend of programming languages for children, Papert saw its greatest potential as an incubator of powerful ideas (Papert, 2000). That is, as a tool to engage children in new ways of thinking and "thinking about thinking" (Papert, 2005). Papert strongly believed in Constructionism, his philosophy of education that argued the power of programming languages became most salient when they provided opportunities for supporting the design and making of personally meaningful computationally-rich projects that invited children to think in new ways (Papert, 1980).

Programming provides an opportunity to engage in logical and abstract thinking, problem solving and the creative design process. In the last decade, new programming languages inspired by LOGO and Constructionism were developed. One of the most popular ones is Scratch (Resnick et. al, 2009), first released in 2007 and available for free, developed by Mitchel Resnick – Papert's disciple. Scratch was designed for children 8 years old and beyond, and provides an icon-based friendly interface, so children can create their own stories, animations and games. Furthermore, in the era of social media, once children create their projects with Scratch they can share them on-line with a community of peers that invites re-mixing. Today, over 29 million projects have been shared on the Scratch website (scratch.mit.edu) and the number keeps growing every day.

Scratch was designed following a "low floors, high ceilings, and wide walls" metaphor (Resnick, 2009). That is, it provides easy ways for novices to get started (low floor), ways for them to work on increasingly sophisticated projects over time (high ceiling) and multiple pathways for engagement for all children with diverse interests (wide walls). However, although the walls are wide, Scratch assumes children have a basic knowledge of reading and writing (programming blocks have words associated with their actions), as well as the ability to manage complexity and the almost infinite possibility of commands. When children are younger than 8 years old, they tend to not have the developmental maturity to use Scratch as they are faced with too many options and too many words.

ScratchJr was created to address this problem. It has even wider walls because it not only engages children with a variety of personal interests, but also with diverse developmental stages (Bers and Resnick, 2015). ScratchJr is a free digital coding playground that introduces powerful ideas of computer science into early childhood education (Bers, 2018b). Just like in the playground, children can choose different activities to do and use their imagination while making projects they care about. In addition, they develop abstract, sequential thinking and problem solving strategies while engaging in computational thinking.

This paper describes ScratchJr, and the decisions made in the early design stages to support young children to become programmers, presents examples of curricular innovations and materials developed for ScratchJr and provides overall analytics data regarding ScratchJr's use in the world, with a focus on Europe.

THE CONTEXT: POLICY CHANGES ABOUT COMPUTER SCIENCE EDUCATION

Along with the design of new programming languages for children, such as ScratchJr, research and policy changes all over the world brought a newfound focus to coding starting in early childhood (Sesame Workshop, 2009; Barron, et al. 2011; International Society for Technology in Education (ISTE), 2007; NAEYC and Fred Rogers Center for Early Learning and Children's Media, 2012; U.S. Department of Education, 2010; K-12 CS framework, https://k12cs.org/). Most of these changes were triggered by the need to educate the workforce of the future automated economy and by the realization that industry was in need of tech savvy workers. For example, the U.S. Bureau of Labor Statistics predicts that employment in information technology occupations will grow 12.5% from 2014 to 2024 (Fayer et al., 2017).

In the US, the "Computer Science for All" initiative was launched to bring programming into every single educational level (Smith, 2016). In Europe, as of the writing of this paper, 16 countries integrate coding in the curriculum at the national, regional, or local level, including: Austria, Bulgaria, the Czech Republic, Denmark, Estonia, France, Hungary, Ireland, Israel, Lithuania, Malta, Spain, Poland, Portugal, Slovakia, and the UK (Balanskat and Engelhardt, 2014; European Schoolnet, 2015; Livingstone, 2012; Bocconi et. al, 2016). Widespread resources are now available in Europe through the European Coding Initiative's resource website, or 'all you need is {C<3DE}', which is akin to the United States' Code.org site (see: allyouneediscode.eu and code.org).

Outside of Europe, countries such as Australia, Singapore, and Malaysia have also established policies and frameworks for introducing computer programming in K-12 education (Australian Curriculum and Assessment Authority, 2015; Digital News Asia, 2015; Malaysia Digital Economy Corporation, 2016; Siu and Mei, 2003). In Argentina, the Ministry of Education's training courses in computational thinking and programming reached more than 10,000 teachers during its first launch (Code.org, 2017) and similar experiences happened in Chile through the Kodea Foundation. Developing countries, such as Ghana, are also introducing coding through non-profit organizations like the Ghana Code Club (Nguyen, 2016).

According to the non-profit organization Code.org, which encourages schools all over the world to adopt programming curricula and promote broad participation in computer science, "The Hour of Code" initiative has surpassed 500 million students served, reaching one out of every 10 students on the planet. This is the largest education campaign in history (Code.org, 2017). However, historically, education campaigns have served a larger role than fulfilling the workforce's demands. They educate the future citizenry of a country. The use of literacy campaigns that mobilize people and resources on a large scale is a long-established practice. H. S. Bhola traces literacy campaigns back to the Protestant Reformation in Europe in the early 1500s (Bhola, 1997). Often, these literacy campaigns supported social, economic, cultural and political reform or transformation. In the 1970s, mass adult literacy campaigns were commonly initiated by governments following liberation wars with a revolutionary or decolonization agenda (Bhola, 1984).

As more people learn to code and computer programming departs the exclusive domain of computer science and becomes central to other professions and to new ways of thinking, coding takes on the civic dimension of literacy (Bers, 2018a). When developing ScratchJr, the goal was to design an introductory programming language not to prepare students for computer science degrees and careers (due to the shortage of programmers and software developers in the industry), but to provide them with the intellectual tools to serve a role in civic society. Coding is more than a technical skill; it is a way to achieve literacy in the 21st century, like reading and writing.

Figure 1. This image shows the programming app interface

Learning to code engages children in new ways of thinking that some researchers have called computational thinking (Wing, 2006; Barr and Stephenson, 2011; International Society for Technology Education and The Computer Science Teachers Association, 2011; Lee et al. 2011). This involves a range of analytical mental tools that are inherent to the field of computer science, including thinking recursively, applying abstraction when figuring out a complex task, and using heuristic reasoning to discover a solution. These mental tools are universally applicable (Wing, 2011). Therefore, they can be taught, not only through Computer Science courses, but in an integrated way with other curricular disciplines at school and from an early age.

Research shows that both from an economic and a developmental standpoint, educational interventions that begin in early childhood have lower costs and durable effects (Cunha and Heckman, 2007). While most nation-wide coding initiatives started targeting older children, there are recent endeavors focusing on early childhood. In Europe, countries such as the United Kingdom have adapted their curriculum to include coding, beginning in early childhood. In Asia, Singapore launched the nationwide PlayMaker initiative that brings robotics, amongst other coding technologies, into early childhood classrooms (Digital News Asia, 2015; Sullivan and Bers, 2017).

However, if the introduction of coding is going to start early, there is a need of technologies and pedagogical approaches that are developmentally appropriate and that take into consideration the cognitive maturity and abilities of young children (Bers, 2018b). ScratchJr was born out of that need.

SCRATCHJR: A DIGITAL PLAYGROUND FOR CODING

ScratchJr is freely available and can be downloaded for use on several platforms including iOS, Android, Amazon tablets, and Chromebooks which are rapidly growing in popularity (Leidl et al., 2017). As of February, 2018, there are over 9.5 million iOS downloads of ScratchJr with an average of 104,000 active users each week and over 20 million projects created. Volunteers from around the world have helped translate ScratchJr into 12 languages.

ScratchJr was developed out of a three-year research grant from the National Science Foundation (DRL-1118664), as a collaboration between the DevTech Research Group at Tufts University, the MIT Lifelong Kindergarten Group, and the Playful Invention Company, and is currently financially supported by the Scratch Foundation. It was released in 2014 and since then its user based has continuously grown (Bers and Resnick, 2015).

Used in classrooms and homes worldwide, ScratchJr enables children, who might or might not know how to read, to create interactive stories and games by snapping together graphical programming blocks. As shown in **Figure 1**, the ScratchJr interface allows children to use blocks that control motion, looks, sound, character communication, and more (Bers, 2018b).

ScratchJr has a palette of programming blocks, a user's library of projects, a main project editor, and tools for selecting and drawing characters and background graphics. Children drag blocks from the palette into the scripting area and then snap them together to create programs that are read and played from left to right. The programming blocks are organized into six categories represented by different colors: yellow Trigger blocks, blue Motion blocks, purple Looks blocks, green Sound blocks, orange Control flow blocks, and red End blocks. When put together as a jigsaw puzzle, these programming blocks allow children to control their character's actions on the screen. The programming blocks span concepts from simple sequencing of motions to more complex control structures.

ScratchJr's design features support problem solving by reducing unnecessary low-level cognitive burdens. These design decisions keep the challenge at an appropriate level and may help young children devote sufficient cognitive resources to the many high-level thinking processes involved in imagining and creating a program.

ScratchJr has a feature called "the grid" that overlays the animation stage (see **Figure 2**). It can be toggled on and off, and is most helpful when used during programming (as opposed to when presenting a project). The grid

Figure 2. This image shows the different ways to program motion using the grid

was designed to help children understand the units of measurement for each programming block for linear movement. The grid is similar to the upper right quadrant of the Cartesian coordinate system, with discrete rather than continuous units of measure. Its numbered axes prompt counting and provide a marker to track counting.

Several design decisions were made for seamless integration with literacy. The ability to create up to four independent "pages" and to integrate text and speech into a project allows children to create their own storybooks with a beginning, middle, and end.

The design and development process of ScratchJr started by observing how young children used Scratch (www.scratch.mit.edu), designed for older children 8 and up, and noting their difficulties (Flannery et al., 2013). For example, we noted that children were getting lost with so many possibilities for programing commands. Thus, we simplified the programming blocks options and offered a more limited programming palette. We also noticed that movement happened too fast and children had a difficult time understanding the relationship between the programming blocks and their resulting actions. We decided to slow down processes, so every block would take time before the triggering of the action.

We worked with hundreds of teachers and children through informal afterschool sessions, educator workshops, experimental classroom interventions, and at-home play sessions. Additionally, we conducted online surveys and face-to-face focus groups to obtain feedback. These provided valuable insights for our design team.

SCRATCHJR'S DESIGN PRINCIPLES

I. Inspired by Bers framework that describes how new technologies for children can become "playgrounds" that encourage open-ended exploration, creativity, imagination and social interactions, as well as skill building, mastering and problem solving (Bers, 2018b), ScratchJr was designed as a digital playground for coding. At the playground, children are exposed to diverse activities to choose from. They can go to the sand box, the swing, the slide, or just run around. They can play with sticks, ride their bikes, or create fantasy worlds. Similarly, while using ScratchJr, children can engage in all kinds of activities beyond coding. They can create and modify characters in the paint editor, record their own voices and sounds, and even insert photos of themselves using the paint editor's camera option. And, of course, they can incorporate those media rich materials into their projects to personalize them. Furthermore, what is unique about a coding playground such as ScratchJr, in contrast to a multimedia creation tool, is that children encounter powerful ideas from computer science when programming their games, animations or stories.

ScratchJr was designed to support children in engaging with seven powerful ideas of computer science that are developmentally appropriate for young children (Bers, 2018b): algorithms, modularity, control structures, representation, hardware/software, design process, and debugging (see **Table 1**). These ideas are aligned with educational computer science frameworks utilized in schools, such as the K-12 Computer Science Framework, the CSTA K-12 Computer Science Standards, and the ISTE Standards for Computer Science Educators.

Table 1. Powerful Ideas from Computer Science and Connections to ScratchJr

Powerful Idea	Definition	Early Childhood Connections	ScratchJr Application
Algorithm	A series of ordered instructional steps taken in a sequence to solve a problem or achieve an end goal.	Understanding abstraction and sequencing.	When children put together the colorful ScratchJr programming blocks in a logical order and create a sequence of actions (a script) for the chosen characters.
Modularity	The breaking down of tasks or procedures into simpler, manageable units that can be combined or re-used to create a more complex process.	Understanding that a complex task needs to be broken down into smaller tasks.	In ScratchJr, a common practice of modularity is to copy a portion of a script (coding sequence) from one character to another. For example, if a child wants to make a ScratchJr dance party featuring several characters, she can create a chunk of code for a dance move and copy it to multiple characters.
Control Structures	Control structures determine the order (or sequence) in which instructions are followed or executed within an algorithm or program. For example, repeat functions, loops, conditionals, events, and nested structures, are all control structures.	Understanding control structures requires an understanding of patterns and the concept of making decisions based on certain conditions as well as cause and effect.	In ScratchJr, children explore the concept of control structures by utilizing control flow blocks that allow them to create loops and repetitions as well as set different variables such as speed.
Representation	Programming languages represent information through the use of a symbol system. At the same time, computers store and manipulate data and values in a variety of ways. In order to be made available, this data is represented in different ways.	Understanding that concepts can be represented using symbols, and that programming languages are formal constructed symbol systems designed to communicate instructions (an algorithm) to a machine.	ScratchJr uses different forms of representations. Colorful blocks represent different types of commands. For example, blue blocks represent motion.
Hardware/Software	Computing systems need hardware and software to operate. The software provides instructions to the hardware, which might or might not be visible. Hardware and software work together as a system to accomplish tasks, such as receiving, processing, and sending information.	Understanding systems and their components, as well as the complex interplay between "instructions" (code) and "objects that receive those instructions".	ScratchJr is the software, the programming language that runs on different hardware devices.
Design process	This iterative process involves several steps: ask, imagine, plan, create, test, improve, and share. The process is open-ended, in that a problem may have many possible solutions.	Understanding that creating a final product to be shared with others involves several steps and continuing revising of the work.	In ScratchJr the design process starts when a child asks a question that gives birth to an idea and ends with creating a final project that can be shared with others. The design process makes computational thinking visible: coding becomes a tool of expression.
Debugging	Fixing problems through systematic analysis and evaluation, while developing troubleshooting strategies.	Learning how to debug is an important skill that is similar to "checking your work in math" or "editing" in literacy. It teaches the powerful lesson that things do not just happen to work on the first try, and that many iterations are usually necessary to get it right.	When using ScratchJr to create a personally meaningful program children naturally engage in debugging by fixing what doesn't work and problem solving.

TEACHING WITH SCRATCHJR

Although ScratchJr is a developmentally appropriate programing language that can be learned without knowing how to read or write, Papert warns against a "technocentric perspective" in which the technology is placed at the center of the teaching and learning process. (Papert, 1987). Therefore, in addition to designing the programming language itself, the ScratchJr team developed teaching materials to support the use of ScratchJr both at home and at school. On the ScratchJr website (scratchjr.org) one can find freely available curriculum units designed for K-2 with a focus on learning ScratchJr by making connections with other content areas such as literacy, narrative genres, and ludic experiences. In addition, ScratchJr Coding Cards (Bers and Sullivan, 2018) have recently been released. This set of cards involves three different kind of activities: off-the-screen games to help children understand the powerful ideas from computer science through unplugged experiences; on-screen activities to learn about the ScratchJr interface; and ScratchJr challenges to build skills on the different aspects of the programming language.

Figure 3. This image shows the Lunar New Year collaborative project

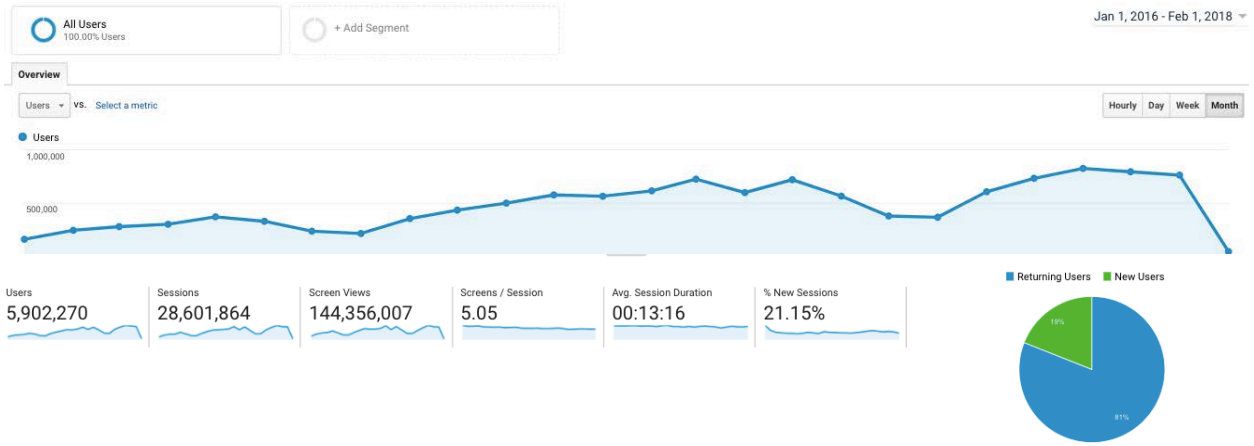


Figure 4. This image shows the Google Analytics overview

Most of the teaching resources developed by the ScratchJr team are focused on helping children engage in computational thinking by making their own personally meaningful projects. Thus, they are designed for one child with one tablet. However, this approach doesn't truly provide a "playground" experience for children. In the playground, children tend to play together. To address this issue, the DevTech research group developed the "Collaborative ScratchJr Projects Guide" to support teachers in the making of multi-tablet collaborative projects, with images and movements that can span across multiple screens and thus involve multiple children. Collaborative ScratchJr projects can have an overall theme, storyline, or learning goal and allow children to interact with the app and with each other in new, creative ways.

In **Figure 3** there is an example project that the DevTech Research Group created. The project features the Lunar New Year, a holiday that is celebrated in many Asian nations to welcome the arrival of spring. Using nine coordinated tablets, this ScratchJr project displays different parts of the festival: the blooming peach and pear flowers, the traditional Dragon Dance, and the lighting of red lanterns and firecrackers. To make this project, programmers had to work together not only through coding, but also by imagining and planning. In the ScratchJr website it is possible to find other examples of collaborative projects.

USING SCRATCHJR: ANALYTICS

Since ScratchJr's launch in 2014 the app has been downloaded over 9.5 million times and is currently being used in every country in the world (except North Korea and Western Sahara). Since January 2016, the ScratchJr team started to collect Google Analytics data to examine usage patterns. Google analytics is a free tool that allows small-to-medium organizations to collect data on user behavior by installing a "cookie" on devices that download ScratchJr. Researchers can gain access to ScratchJr user activity as it happens in real-time on the app, as well as audience demographics, acquisition, and behavior (Leidl et al., 2017).

An overview of the type of data acquired through Google Analytics can be seen in **Figure 4**. Weekly usage in **Figure 5** shows a tall peak occurring during Computer Science Education Week each year in December. Thus, this educational initiative is widely successful in promoting coding in schools. Furthermore, there tends to be a sharp decline in usage from the end of December to January, likely indicating that while students are home for winter vacation they are not using ScratchJr frequently. There is also a dip in usage during the summer months. The data indicate that ScratchJr is used mainly in school settings.

Figure 5. This line-graph shows weekly ScratchJr usage

During the two-year period from January 2016 to February 2018, over 20 million new projects were created and over 26 million existing projects were edited and revised, showing that ScratchJr users are actively working on improving and testing their programs. Of these projects, over 7 million have been created in Europe (35%) and over 9 million have been edited and revised in Europe (45%).

Furthermore, over 600,000 projects have been shared with others via email or Apple AirDrop®. Of these projects, 225,000 have been shared in Europe (38%). In this relatively short amount of time, over 406 million programming blocks have been used, the five most common blocks being "Forward," "Start on Green Flag", "Up", "Back", and "Say" and the three least common blocks being "Stop", "Start on Touch" and "Start on Message". These results from Google Analytics are consistent with previous research on coding and cognitive development (Flannery and Bers, 2013; Portelance et al., 2015). Many of the most popular blocks for children to use are the blue motion blocks, which are simple commands for children to start programming their characters. However, the least common blocks include starting a program (besides on green flag) and sending messages, which are cognitively more challenging concepts for children to grasp as they require a higher level of sequencing abilities.

The frequency of block usage in Europe reflects how ScratchJr is used internationally. The consistency among users of various continents is likely due to the fact that children tend to code in a similar manner regardless of their location, and ScratchJr's freely available curriculum shares many similarities.

Additionally, ScratchJr maintains a rate of 249,000 returning users each month, while still attaining a consistent rate of 255,000 new users each month. In Europe there is a rate of 50,000 returning users each month with 83,000 new users per month. Data usage in Europe reflects the international trend of having more new users each month than returning users thus showing that ScratchJr is a consistently growing app.

According to Google Analytics the geographical areas with the most ScratchJr usage is the "Americas" (consisting of North America, South America, Central America, and the Caribbean) with 43% of the total usage. Europe is the next continent with the most usage with 34%, followed by Asia (12%), Oceania (consisting of Australia, New Zealand, and Polynesia) (9%), and Africa (1%).

In Europe the countries with the most ScratchJr usage are the United Kingdom (40% of European usage), Sweden (10%), France (10%), Spain (7%), Italy (5%), Finland (5%), the Netherlands (4%), Poland (3%), Germany (2%), and Denmark (2%). It is important to note, however, that Google Analytics only takes into account the total number of users for each country and that the percentages are not proportional to the population of the countries.

Figure 6 provides visual data for the percentage of ScratchJr usage by each country's population. Taking population into account, the top countries using ScratchJr are Sweden (2% of the population), Finland (2%), the United Kingdom (1%), and Denmark (1%). This should come as no surprise as the UK and Nordic countries have robust initiatives to integrate coding into the curriculum.

According to EU Code Week the "British government wants to ensure that all pupils can understand and apply the fundamental principles and concepts of computer science." Furthermore, coding in the UK is seen as a means of creative thinking, not just filling an industry gap. In primary schools 5-7 year olds learn what algorithms are and how they are executed on digital devices, as well as about creating and debugging programs as well as using logical reasoning to predict the outcome of a script (EU Code Week 2014). In Denmark and Sweden the concepts of Computational Thinking are taught to children including abstraction and debugging as well as digital citizenship. In Finland algorithmic thinking plays a role, however it is introduced mainly in mathematics. Furthermore, Professional Development Programs in Finland and Sweden introduce primary school educators to ScratchJr (Bocconi et al., 2018).

Figure 7 shows the total ScratchJr usage in each country in Europe. Compared to the rest of the world, countries in Europe have very high ScratchJr usage. In **Figure 8** the top ten countries using ScratchJr across the world are shown. The United Kingdom is the second country in terms of usage (18% of international ScratchJr users). Other top countries include Sweden (5%), France (4%), and Spain (3%).

ScratchJr usage in Europe based on day of the week and month reflects international trends. In **Figure 9** the ScratchJr usage in Europe based on day of the week is shown. The most popular days of the week are Friday and Thursday, with the least popular being Saturday and Sunday. **Figure 10** visualizes the ScratchJr usage in Europe

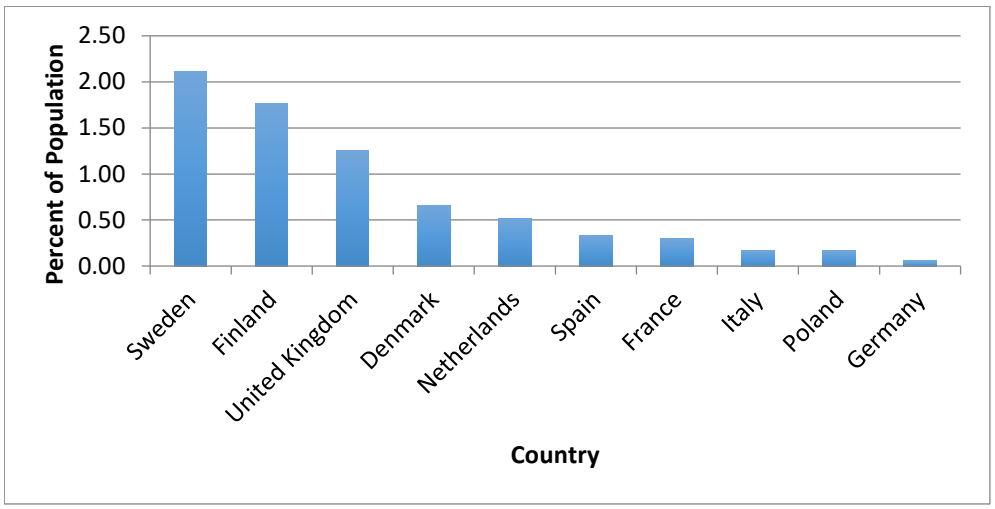


Figure 6. This chart shows the percentage of the population using ScratchJr

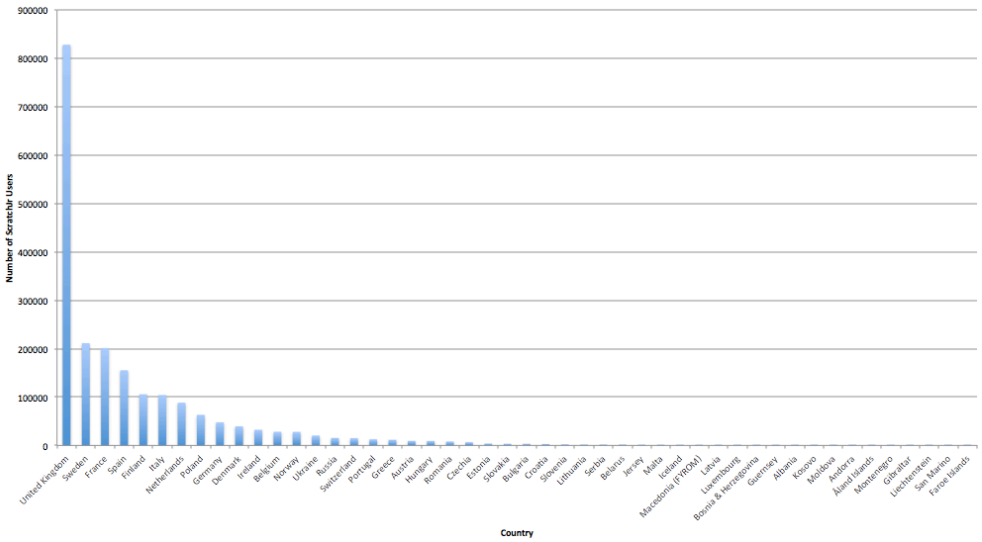


Figure 7. ScratchJr users for each country in Europe

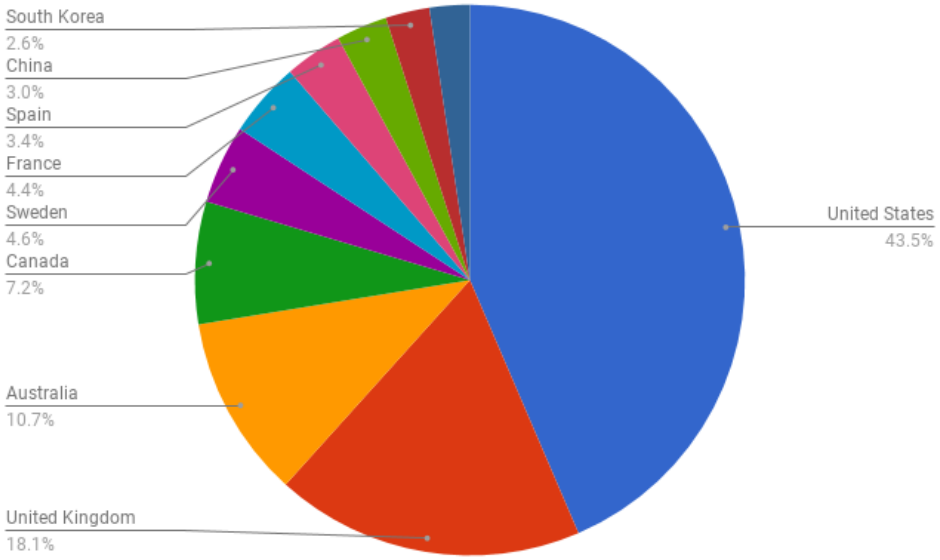


Figure 8. The top 10 countries in the world using ScratchJr

by month of the year. The most active months are November and December while the least active are July and August, vacation time. The increase in December is most likely due to Computer Science Education week initiatives.

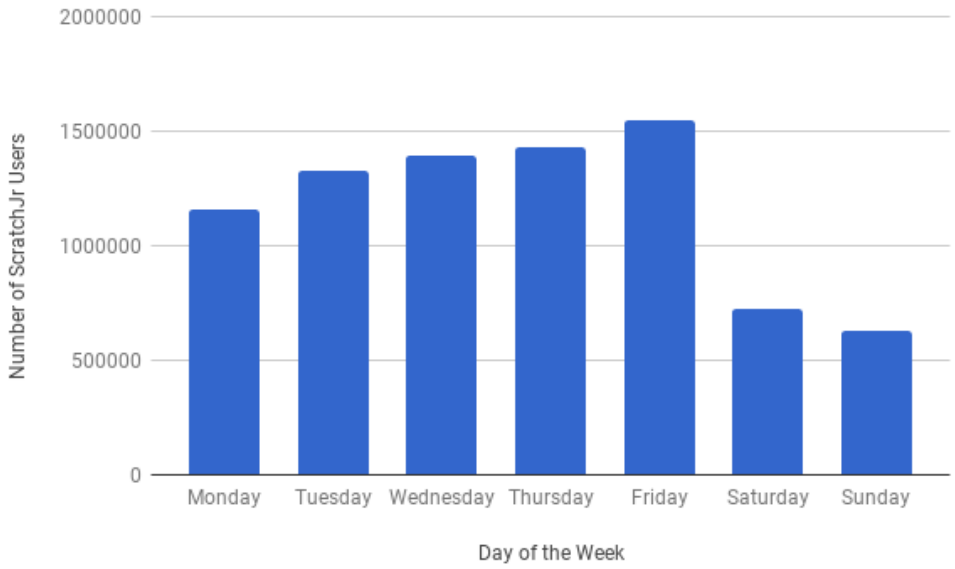


Figure 9. Shows the number of ScratchJr users in Europe by day of the week

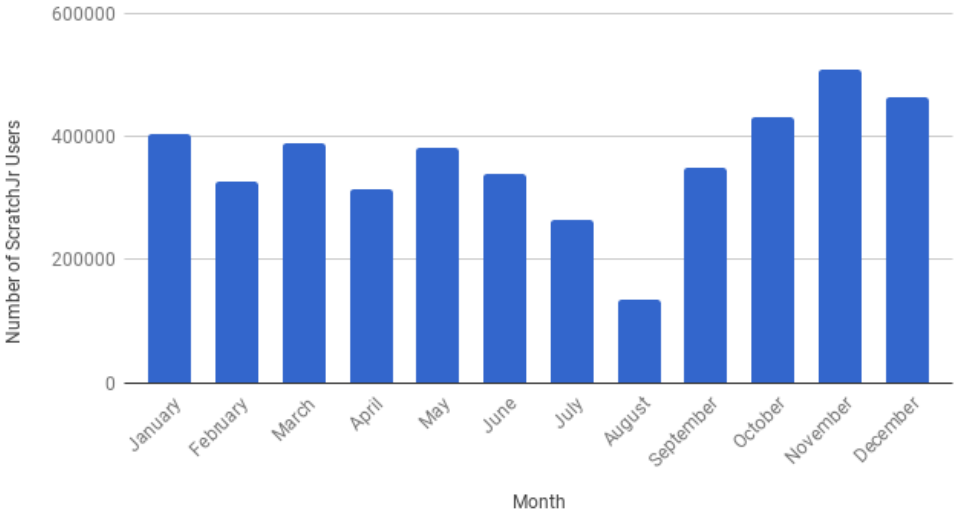
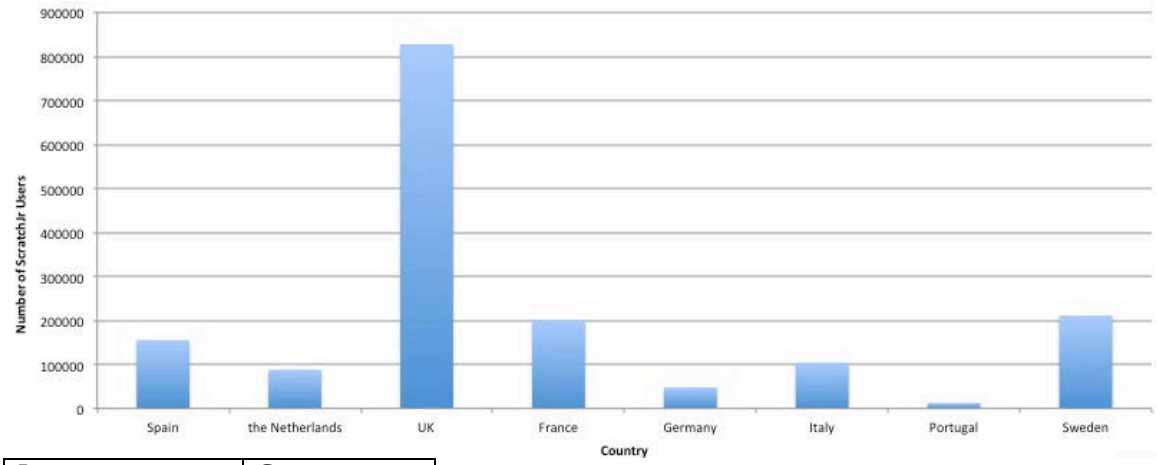


Figure 10. Shows the number of ScratchJr Users in Europe by month

Google Analytics tracks language by recording a user's language setting from their browser using ISO codes. Percentages reflect the proportion of devices registered with a particular language with all registered languages. The most popular registered languages in Europe are:

1. English-Great Britain (32%)
2. Swedish-Sweden (8%)
3. French-France (7%)
4. Spanish-Spain (4%)
5. English-US (4%)
6. Finnish-Finland (4%)
7. Italian-Italy (3%)
8. Dutch-Netherlands (3%)
9. Polish-Poland (2%)
10. German-Germany (1%)

The ScratchJr team places a priority on trying to support language localization. Therefore, ScratchJr volunteers from around the globe use Transifex, a web-based translation platform, to translate the ScratchJr app and website. Volunteers can request to localize in a certain language and are added to that team's page. They can then translate strings, a sequence of text, for the app and website. For the website, volunteers have the option of translating "live", thus seeing how their translations look on the ScratchJr webpage. After the strings are translated certain volunteers who have been promoted to reviewers look over the translations to ensure their accuracy. Currently



Language	Country
Catalan and Spanish	Spain
Dutch	the Netherlands
English	UK
French	France
German	Germany
Italian	Italy
Portugese	Portugal
Swedish	Sweden

Figure 11. The number of ScratchJr users in European countries with languages included in ScratchJr app

ScratchJr is translated into Catalan, Chinese, Dutch, French, English, German, Italian, Japanese, Portugese, Spanish, Swedish, and Thai. Eighteen additional languages are currently being translated to be included in future updates. One caveat to language translation is that only languages that are written left-to-right can be included. Right-to-left languages such as Hebrew and Arabic cannot be translated for ScratchJr because of how the app was initially configured. In addition to the ScratchJr app and website, the Official ScratchJr Book, written by Dr. Marina Umaschi Bers and Dr. Mitchel Resnick (Bers and Resnick, 2015), has been translated into Dutch, Swedish, Korean, Chinese, French, Turkish, Polish, Chinese, and Spanish.

Of the twelve languages that the ScratchJr app supports, ten are the primary language spoken by countries in Europe. **Figure 11** shows the ScratchJr usage in these ten countries (Spain includes both Spanish and Catalan). The relationship between translated language and ScratchJr usage is two-fold. Many of the languages that are included in the latest ScratchJr update are from countries which showed high ScratchJr usage before the translated version of the app was introduced. Having a high demand for translation encourages volunteers to translate ScratchJr into their language of origin.

CONCLUSIONS

In accordance with the international growing trend, the teaching of coding is becoming an increasingly important focus in European education. This paper describes ScratchJr, the most popular programming language for early childhood, and presents an overview of how it is being used in Europe. The paper shows that European usage trends are in alignment with the rest of the world in terms of coding patterns and daily and monthly usage. For example, students in Europe are more likely to use ScratchJr during the school week than the weekend, showing that the app is used heavily in educational settings. Furthermore, countries with stronger policies regarding the teaching of computer science, such as the UK and the Nordic countries, show higher usage.

Additionally, of the twelve languages that the ScratchJr app supports, ten are the primary languages spoken in European countries. Future work will focus on expansive localization and translation initiatives, informed by the analytics data.

ACKNOWLEDGEMENTS

The work on ScratchJr is possible through a generous grant from the National Science Foundation (NSF DRL-1118664) and the Scratch Foundation. I am thankful to the many teachers and parents who contributed by completing the ScratchJr survey and the ScratchJr team both at the DevTech research group at Tufts University

and at the MIT Media Lab. In addition, I am thankful to Melissa Viezel for her work on analysis of data for this paper and Amanda Strawhacker and Amanda Sullivan for their contributions to the writing.

REFERENCES

Allan, V., Barr, V., Brylow, D. and Hambrusch, S. (2010, March). Computational thinking in high school courses. *In Proceedings of the 41st ACM technical symposium on Computer science education* (pp. 390-391). ACM. https://doi.org/10.1145/1734263.1734395

Australian Curriculum and Assessment Authority. (2015). Information and Communication Technology (ICT) capability. Available at: https://www.australiancurriculum.edu.au/f-10-curriculum/general-capabilities/information-and-communication-technology-ict capability/?searchTerm=2015+technology+and+computer+programming#dimension-content (Accessed 28 March 2018).

Balanskat, A. and Engelhardt, K. (2014): Computing our future: computer programming and coding - priorities, school curricula, and initiatives across Europe. European Schoolnet. Available at: http://www.eun.org/c/document_library/get_file?uuid=521cb928-6ec4-4a86-b522-9d8fd5cf60ce&groupId=43887 (Accessed 28 March 2018).

Barr, V. and Stephenson, C. (2011). Bringing computational thinking to K-12: What is involved and what is the role of the computer science education community? *ACM Inroads,* 2(1), 48–54. https://doi.org/10.1145/1929887.1929905

Barron, B., Cayton-Hodges, G., Bofferding, L., Copple, C., Darling-Hammond, L. and Levine, M. (2011). *Take a giant step: a blueprint for teaching children in a digital age*. New York: The Joan Ganz Cooney Center at Sesame Workshop.

Bers, M. U. and Resnick, M. (2015). The official ScratchJr book. San Francisco, CA: No Starch Press.

Bers, M. U. (2010). The TangibleK tobotics program: applied computational thinking for young children. *Early Childhood Research and Practice*, 12(2). Available at: http://ecrp.uiuc.edu/v12n2/bers.html. (Accessed 28 March 2018).

Bers, M. U. (2018a). *Coding as a literacy for the 21st Century*. Education Week.

Bers, M. U. (2018b). *Coding as a Playground*. London and New York: Routledge Press.

Bers, M. U. and Sullivan, A. (2018, July). *ScratchJr Coding Cards: Creative coding activities*. San Francisco, CA: No Starch Press. ISBN-13: 978-1-59327-899-1.

Bhola, H. S. (1997). Systems thinking, literacy practice: Planning a literacy campaign in Egypt. *Entrepreneurship, innovation and change*, 5(1), 33-47.

Bhola, H. S. (1984). *Campaigning for Literacy: Eight National Experiences of the Twentieth Century, with a Memorandum to Decision-Makers*. UNIPUB, 4611-F Assembly Drive, Lanham, MD 20703.

Bocconi, S., Chioccariello, A. and Earp, J. (2018). The Nordic approach to introducing Computational Thinking and programming in compulsory education. *Report prepared for the Nordic@BETT2018 Steering Group*. Available at: https://doi.org/10.17471/54007 (Accessed 28 March 2018).

Bocconi, S., Chioccariello, A., Dettori, G., Ferrari, A., Engelhardt, K., Kampylis, P. and Punie, Y. (2016). Developing computational thinking in compulsory education. *European Commission, JRC Science for Policy Report.*

Brennan, K. and Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. *Proceedings of the 2012 Annual Meeting of the American Educational Research Association*, Vancouver, Canada. Available at: http://web.media.mit.edu/~kbrennan/files/Brennan_Resnick_AERA2012_CT.pdf (Accessed 28 March 2018).

Clements, D. H. (1985). Differential effects of computer programming (Logo) and computer assisted instruction on young children's executive processes and cognitive development [Summary]. In *51th Biennial Meeting of the Society for Research in Child Development,* 5, 59. https://doi.org/10.2190/RCNV-2HYF-60CM-K7K7

Clements, D. H. (1987). Longitudinal study of the effects of Logo programming on cognitive abilities and achievement. *Journal of Educational Computing Research*, 3(1), 73-94.

Clements, D. H. and Meredith, J. S. (1993). Research on Logo: Effects and efficacy. *Journal of Computing in Childhood Education*, 4(4), 263-290.

Code.org. (2017). Computer science pledges and announcements. Available at: https://medium.com/@codeorg/computer-science-pledges-and-announcements-8f6aaf33239e (Accessed 28 March 2018).

Cunha, F. and Heckman, J. (2007). The technology of skill formation. *American Economic Review,* 92(2), 31-47. https://doi.org/10.1257/aer.97.2.31

Digital news Asia. (2015). IDA launches $1.5m pilot to roll out tech toys for preschoolers. Available at: https://www.digitalnewsasia.com/digital-economy/ida-launches-pilot-to-roll-out-tech-toys-for-preschoolers. (Accessed 28 March 2018).

Ertas, A. and Jones, J. C. (1996). *The engineering design process* (2nd ed.). New York, NY: John Wiley and Sons, Inc.

European Schoolnet (2015). *Computing our Future: Computer Programming and Coding Priorities, School Curricula, and Initiatives across Europe.*

Fayer, S., Lacey, A. and Watson, A. (2017, January). STEM occupations: Past, present, and future. *Spotlight on Statistics. U.S. Bureau of Labor Statistics.*

Flannery, L. P. and Bers, M. U. (2013). Let's dance the "robot hokey-pokey!": children's programming approaches and achievement throughout early cognitive development. *Journal of Research on Technology in Education,* 46(1), 81-101. https://doi.org/10.1080/15391523.2013.10782614

Flannery, L. P., Kazakoff, E. R., Bontá, P., Silverman, B., Bers, M. U., and Resnick, M. (2013). Designing ScratchJr: support for early childhood learning through computer programming. *In proceedings of the 12th international conference on interaction design and children* (IDC '13). ACM, New York, NY, USA, 1-10. https://doi.org/10.1145/2485760.2485785

International Society for Technology in Education. (2007). NETS for students 2007 profiles. Washington, DC: ISTE. Available at: www.iste.org/standards/nets-for-students/nets-for-students-2007-profiles.aspx#PK-2. (Accessed 28 March 2018).

International Society for Technology in Education and the Computer Science Teachers Association. (2011). Operational definition of computational thinking for K-12 thinking. *International Society for Technology in Education (ISTE) and the Computer Science Teachers Association (CSTA).* Available at: http://www.iste.org/docs/ct-documents/computational-thinking-operational-definition-flyer.pdf. (Accessed 28 March 2018).

K-12 Computer Science Framework Steering Committee. (2016).

Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W., Erickson, J., et al. (2011). Computational thinking for youth in practice. *ACM Inroads,* 2(1), 32–37. https://doi.org/10.1145/1929887.1929902

Leidl, K., Bers, M. U., Mihm, C. (2017). Programming with ScratchJr: a review of the first year of user analytics. In the proceedings of the *International Conference on Computational Thinking Education*, 2017. Wanchai, Hong Kong.

Livingstone, S. (2012). Critical reflections on the benefits of ICT in education. *Oxford Review of Education,* 38(1), 9-24. ISSN 0305-4985. https://doi.org/10.1080/03054985.2011.577938

Malaysia Digital Economy Corporation (2016). Creating a nation of digital makers key to Malaysia's future success. Available at: https://www.mdec.my/news/mydigitalmaker. (Accessed 28 March 2018).

Massachusetts Department of Elementary and Secondary Education. (2016). Massachusetts Science and Technology/Engineering Curriculum Framework.

NAEYC and Fred Rogers Center for Early Learning and Children's Media. (2012). Technology and interactive media as tools in early childhood programs serving children from birth through age 8. Joint position statement. Washington, DC: NAEYC. Latrobe, PA: Fred Rogers Center for Early Learning at Saint Vincent College. Available at: www.naeyc.org/files/naeyc/file/positions/PS_technology_WEB2.pdf. (Accessed 28 March 2018).

National Academies of Science. (2010). Report of a workshop on the scope and nature of computational thinking. Washington DC: National Academies Press.

Nguyen, M. (2016). Turning Ghana Upside Down with Coding. Available at: https://medium.com/scratchteam-blog/turning-ghana-upside-down-with-coding-3811c4777e4b. (Accessed 28 March 2018).

Papert, S. (1980). Mindstorms: Children, Computers, and Powerful Ideas, Basic Books, Inc. https://doi.org/10.3102/0013189X016001022

Papert, S. (1987). Information technology and education: Computer criticism vs. technocentric thinking. *Educational researcher*, 16(1), 22-30. https://doi.org/10.1147/sj.393.0720

Papert, S. (2000). What's the big idea? Toward a pedagogy of idea power. *IBM Systems Journal,* 39(3.4), 720-729.

Papert, S. (2005). You can't think about thinking without thinking about thinking about something. *Contemporary Issues in Technology and Teacher Education,* 5(3/4), 366-367.

Portelance, D. J., Strawhacker, A. and Bers, M. U. (2015). Constructing the ScratchJr programming language in the early childhood classroom. *International Journal of Technology and Design Education*, 1-16. Online First.

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., ... and Kafai, Y. (2009). Scratch: programming for all. *Communications of the ACM*, *52*(11), 60-67. https://doi.org/10.1145/1592761.1592779

Sesame Workshop. (2009). Sesame workshop and the PNC Foundation join White House effort on STEM education. Available at: http://www.sesameworkshop.org/newsandevents/pressreleases/stemeducation_11212009 (Accessed 28 March 2018).

Siu, K. W. M. and Mei, S. L. (2003). Technology Education in Hong Kong: International Implications for Implementing the "Eight Cs" in the Early Childhood Curriculum. *Early Childhood Education Journal*, 31. 143-150. https://doi.org/10.1023/B:ECEJ.0000005315.91625.dc

Smith, M. (2016). Computer Science for all. The White House Blog. Available at: https://obamawhitehouse.archives.gov/blog/2016/01/30/computer-science-all. (Accessed 28 March 2018).

Sullivan, A. and Bers, M. U. (2017). Dancing robots: Integrating art, music, and robotics in Singapore's early childhood centers. *International Journal of Technology and Design Education*. Online First. https://doi.org/10.1007/s10798-017-9397-0

The Logo Foundation. (2015). Logo History. Available at: http://el.media.mit.edu/logo-foundation/what_is_logo/history.html (Accessed 28 March 2018).

U.S. Department of Education, Office of Educational Technology. (2010). Transforming American education: Learning powered by technology. Washington, DC: U.S. Department of Education, Office of Educational Technology. Available at: http://www.ed.gov/technology/netp-2010 (Accessed 28 March 2018).

Wing, J. (2006). Computational thinking. *Communications of the ACM,* 49(3), 33–36. https://doi.org/10.1145/1118178.1118215

Wing, J. (2011). Research notebook: Computational thinking—What and why? The Link Magazine, Spring. Carnegie Mellon University, Pittsburgh. Available at: https://www.cs.cmu.edu/link/research-notebook-computational-thinking-what-and-why (Accessed 28 March 2018).

