EVALUATION OF EFFORTS TO EXPOSE MIDDLE SCHOOL STUDENTS TO COMPUTATIONAL THINKING: A REPORT ON THE COSMIC PROGRAM

A Thesis
by
KARA ELISE BEASON

Submitted to the School of Graduate Studies
at Appalachian State University
in partial fulfillment of the requirements for the degree of
MASTER OF SCIENCE

December 2019
Department of Computer Science

EVALUATION OF EFFORTS TO EXPOSE MIDDLE SCHOOL STUDENTS TO COMPUTATIONAL THINKING: A REPORT ON THE COSMIC PROGRAM

A Thesis
by
KARA ELISE BEASON
December 2019

APPROVED BY:

_____
Dr. James B. Fenwick Jr.
Chairperson, Thesis Committee

_____
Dr. Cindy Norris
Member, Thesis Committee

_____
Dr. Alice McRae
Member, Thesis Committee

_____
Dr. Rahman Tashakkori
Chairperson, Department of Computer Science

_____
Mike McKenzie, Ph.D.
Dean, Cratis D. Williams School of Graduate Studies

Abstract

EVALUATION OF EFFORTS TO EXPOSE MIDDLE SCHOOL STUDENTS TO COMPUTATIONAL THINKING: A REPORT ON THE COSMIC PROGRAM

Kara Elise Beason
B.S., Florida State University
M.S., Appalachian State University

Chairperson: Dr. James B. Fenwick Jr.

Computational thinking (CT) is a set of concepts and problem solving skills that are not only imperative for computer scientists, but important and applicable to nearly every discipline. In the past decade, many efforts have been made to develop and evaluate CT in primary and secondary students. This push for CT development in students seeks to prepare their problem-solving skills for a world where technology is ubiquitous, as well as to understand and mitigate the underrepresentation of women and minorities in STEM careers through exposure to computer science early on. COSMIC is one such effort that took place in Caldwell County middle schools from 2015 through 2017. The COSMIC program was created and supervised by researchers at Appalachian State University who supported teachers in hosting after school clubs and summer camps to teach students CT concepts through the use of the CS First curriculum and Scratch programming language. This thesis analyzes the impact of COSMIC using a mixed-mode approach of quantitative and qualitative data. The COSMIC effort was successful in its efforts to improve student awareness, knowledge, and skill of CT concepts, perspectives, and practices.

Acknowledgements

First, I would like to thank my advisor Dr. Jay Fenwick. His passion for education and inclusivity is inspirational, and the work he has done through the COSMIC program has touched the lives of so many. Without his constant support and encouragement, I probably would not have made it through my master's degree. I would also like to thank the members of my thesis committee, Dr. Cindy Norris and Dr. Alice McRae. Thank you not only for believing in me, but for being two amazing role models for women like me in computer science.

I would also like to thank the Caldwell county school teachers who are working every day to improve the lives of future generations. It was a pleasure to meet and work with and learn from so many passionate and caring individuals. I'd also like to thank all of the delightfully smart and kind middle school students I was fortunate enough to work with through COSMIC club. Without them this thesis would not have been possible.

Next, I would like to thank Cassidy Baker, Ethan Fenwick, and Tyler Stein for the countless hours they spent doing data entry, as well as running Dr. Scratch on hundreds of projects. Their hard work and thorough analysis contributed greatly to the work in this thesis and made my life a whole lot easier.

Finally, I would like to thank my friends and family, whose unending emotional support was vital throughout my master's degree and the writing of this thesis. A master's degree may technically be awarded to only one person, but it truly takes a village of support. I am infinitely grateful.

Table of Contents

List of Tables

List of Figures

Chapter 1- Introduction

Computational thinking (CT) is an approach to solving problems, designing systems, and understanding human behavior that draws on concepts fundamental to computing [23]. A core concept of CT is the use of abstraction to solve problems. This application of analytical thinking using abstractions to solve problems is not specific to the field of computer science; it is a cross-disciplinary concept. However, in computer science, one's ability to abstract is imperative. Engineers use computational thinking to design large complex systems that operate within the constraints of the real world; and, mathematicians use computational thinking to efficiently solve mathematical problems that would have previously taken an exorbitant amount of time [23]. The application of mathematical approaches that rely on computation, such as Markov Chain Monte Carlo methods, have provided an avenue of exploration previously unachievable in numerous diverse fields [21]. The mental "tool-kit" of an efficient computational thinker is applicable to nearly all disciplines.

Given the wide applicability and value of CT, many educators are trying to add computational thinking to the core curricula of elementary through high school education programs. Middle school students, while old enough to begin understanding thinking in a computational way, are often not provided with the resources or opportunities to develop a strong basis in computational thinking. Most middle school students have probably used a computer as technology is used in the classroom as a medium for teaching, but rarely are kids exposed to computer programming within the standard K-12 curriculum. And it is well known that computer programming involves substantial computational thinking skills. The use and development of computational thinking skills not only enriches students' relationship to a rapidly growing and increasingly integrated technological world

around them, it also serves to enrich their understanding of other disciplines already considered core curricula [21]. Moreover, exposure to computational thinking concepts before high school also serves to minimize the underrepresentation of women and minorities in computational fields [21]. To mitigate the lack of incorporation of computer science courses into regular curriculum, after-school programs and summer camps have been offered to expose young students to coding, computer science, and computational thinking in general. In these programs, young students are taught basic computational thinking concepts and practices employed in software development.

The goal of this research is to evaluate and present the results of one such after-school club, "COSMIC: Computer Science for Middle Schools in Caldwell County: A Pilot Approach to Targeting Underrepresented Middle School Students in Computer Science." This program was offered in four middle schools over a three year time period to engage middle school students in computational thinking. COSMIC used the visual programming language Scratch in conjunction with the CS First curriculum developed by Google. Scratch is used in many educational programs as a means of teaching kids programming in a more fun and consumable way than a "standard" introductory computer science course that one may find at the high school or college level. As part of a grant and in association with Appalachian State University, this research aided in teaching an after-school program using the CS First (Google) curriculum and Scratch by placing a computer science graduate student at each meeting of the club. The goal of the COSMIC club was to teach kids programming concepts and practices, in hopes that this exposure to computer science at an early age will instill a positive relationship with the technology in the world around them. The COSMIC efforts also used, in a limited way, a couple of other programming tools: CodeHS and SL Nova. However, the data gathered and focus of this thesis are the club efforts using Scratch and CS First.

This research joins a larger aggregation of work showing that kids do indeed learn programming concepts and practices through after-school programs or clubs, and this learning can be assessed through the methodology of surveys, field notes, and project analysis [4,7,16–18]. This work will present a multifaceted and detailed analysis of the results of the COSMIC program, as well as conclusions drawn from this research. The analysis of these results aims to fortify the assertion made by many such programs: that computer science concepts can be taught to middle school aged students with quantifiably positive results, that COSMIC and similar programs make a positive impact on the participants' perception of computer science concepts, and that this incorporation of computer science is not only beneficial to those being taught but is an educational right of the 21st century [11].

The next chapter provides background on computational thinking, the CS First curriculum and Scratch tool, and the COSMIC program structure. Chapter 3 describes related research. Chapter 4 describes data collection methods and sources with Chapter 5 explaining the results of data collection and analysis. Chapter 6 summarizes, concludes future work, and presents ideas for future work.

Chapter 2 - Background

It is important that a working definition for computational thinking (CT) as well as the curricula used in conducting the research is established and understood. These concepts are the scaffolding on which our results are qualified. This research focuses on Scratch as the main vehicle for teaching computational thinking to middle schoolers. Scratch is a visual programming language created by MIT, aimed at teaching kids K-12 introductory to advanced programming concepts. In conjunction with the Scratch programming language, Google's CS First curriculum is implemented in these after-school programs. CS First is designed to aid teachers and volunteers in providing a structured curriculum for their computer science programs.

2.1 Computational Thinking

There are many studies in existence that seek to teach and quantify computational thinking. In March 2006, *Communications of the ACM* published an article written by Jeanette Wing in it's Viewpoint section in which Wing suggested that Computational Thinking (CT) is a universally applicable skill set instead of just something necessary for computer scientists [22]. Though the idea has been around for decades, this helped spark a resurgence in the focus on CT's importance in K-12 education. This interest culminated in the inception of many programs in the 2000s that sought not only to fund CT research and development for K-12 schools, but to prepare the teachers who would be able to facilitate this directive [6].

Computational thinking definitions share some universal ideas, such as the assertion that computing is a creative activity. There is also a focus on the concept of algorithms and abstraction

as tools to facilitate and express solutions to computational problems. Perhaps most importantly, there is agreement on the application of computational thinking concepts and practices to cross-disciplinary problems. In 2012, the Royal Society (London's society for improving natural knowledge) defined CT as "...the process of recognising aspects of computation in the world that surrounds us, and applying tools and techniques from Computer Science to understand and reason about both natural and artificial systems and processes" [6]. This research will use the definition set forth by Brennan and Resnick in their 2012 research "New frameworks for studying and assessing the development of computational thinking" [3]. This definition uses Scratch as the platform to define and discuss their definition of CT, making it easy to translate and apply their definition to COSMIC's research with Scratch and CT. They assert that computational thinking is multi-faceted, consisting of three main sub-categories: concepts, practices, and perspectives.

Computational thinking concepts consist of logic and control flow, conditionals, loops, user interaction, parallelism, and mathematical concepts used widely in programming which transfer to other programming and non-programming contexts [3]. These concepts are defined and mapped to Scratch blocks and concepts as such:

- Sequences: The expression of a task through a series of instructions or individual steps that are executed by the computer.
- Loops: Expressing repeating instructions more succinctly rather than explicitly writing out each step, such as enclosing the step in a "repeat [4]" block in Scratch rather than using the same block 4 times in a linear sequence.
- Events: When one thing happens that causes another thing to happen, such as the pressing of a key or the clicking of a sprite.

- Parallelism: Different sequences of instructions happening at the same time. For instance, a character being programmed to dance for 10 seconds and speak a sentence simultaneously when the green flag is clicked.
- Conditionals: Outcomes that can be different based on the meeting of one or more conditions. For instance, a character might be set to walk until it is touching another character, then stop.
- Operators: Symbols that allow the user to perform logical or mathematical operations, like adding two numbers, or concatenating a string together.
- Data: Values that are stored, manipulated, and retrieved. Scratch has two data types: variables and lists.

These CT concepts are clearly the easiest to quantify in a program provided one knows and understands a programming language. CT concepts alone however do not paint a complete picture of a computational thinker. They are concepts that a computational thinker uses to approach and solve a problem. Computational thinking practices are more complex, including practices such as modularization, abstraction, testing and debugging, soliciting community help in solving problems, as well as "remixing" others' projects. Remixing is a term used in the Scratch community to describe taking someone else's code base and changing or adding on to it [3]. While "remixing" is a Scratch-specific term, the practice of reusing code and building upon the work of others is prevalent in Computer Science. An iterative approach to problem solving is also an important CT practice, wherein one learns to be adaptive when faced with changing plans and processes. While some practices like testing and debugging are easier to quantify by analyzing a project (Does it work? Can I break it?), often CT practices are observed during the development process.

Even more complex are computational thinking perspectives. Brennan and Resnick describe these perspectives as the way one relates to the technological world around them. Computational perspectives describe the nature of a person's technological activities and relationship to the technology they use. Although a person may use technology frequently and well, the nature of this use may be consumerist. A computational thinker sees computation as more than something to consume: it's something to be leveraged to create and express themselves [3]. This is indicative of an active relationship with technology.

Another important aspect of computational thinking perspectives is connecting. Creating and learning are social practices that are greatly enriched by participation in the community and interaction with others. Scratch aides this perspective by providing a large online community with several avenues of communication possible. Students can create for others, view what others are creating, and get help with questions they may have. This makes creating programs more purposeful and engaging for the students and encourages continued participation [3].

Questioning is another integral part of computational thinking perspectives. Questioning describes a person's connection with the technology surrounding them every day. Computational thinkers understand the technology in their world or know that they can come to understand it through questioning. They are not passive users of technology they do not understand; instead, they feel confident in their ability to navigate the technological world around them.

2.2 Scratch

Scratch is a free, visual programming language created as part of the Lifelong Kindergarten Group at the MIT Media Lab. Scratch is designed to teach kids to think creatively, reason systematically, and work collaboratively [24]. Scratch allows kids to learn programming concepts in

a creative and fun way. They can create games, cartoon animations, music videos, tutorials, and more.

The Scratch programming environment was designed with an emphasis on easy navigation. It's comprised of a browser-based, single-window interface with multiple panes as shown in Figure 2.1. When creating a project, the user can see the project stage in the upper right quadrant of the window. This is where their program behavior is visibly observed. Under this is the sprite pane that identifies all the interactive "characters" of the project. The script pane takes up the large space in the center of the window and contains all of the logic for the program. Finally, the toolkit pane is on the left side of the window and is where the user can select different action blocks.. Figure 2.1 contains an example of what the user sees when working on a Scratch project. The "stage" is on the upper right, multiple scripts can be seen in the scripting environment in the center, the tools palette is the left vertical pane, and the interactive "character" sprites are in the lower right.

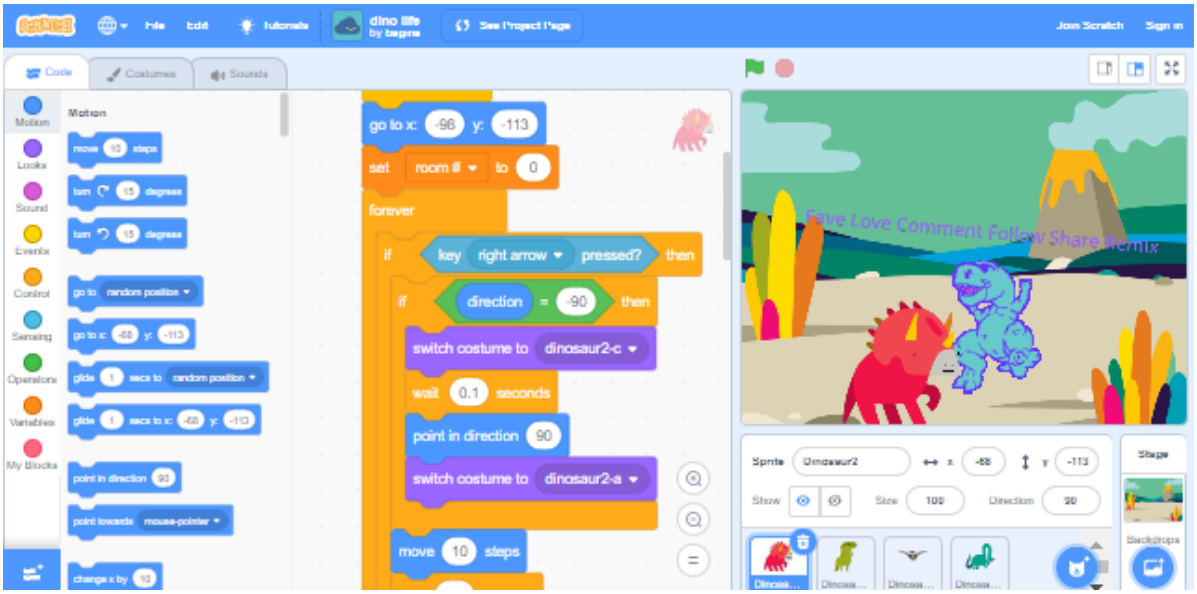


Figure 2.1 Scratch Panel

Scratch allows users to drag and drop color-coded "blocks" of varying categories to create scripts that can run simultaneously. Scratch treats "sprites" as individual agents that can be assigned behavior through scripts. Users can dictate the behavior of sprites and the stage using conditionals (if statements, "when [object or button] clicked" blocks), for and while loops, creating and storing variables and lists, and many other aspects of the logic and control flow seen in "traditional" programming languages and environments. Scratch is unique in that the dragging and dropping of blocks prevents users from making syntax errors. If a user tries to use two acceptable blocks together, the blocks will "snap" into place. However, if the blocks do not go together, such as trying to drag a "wait 3 seconds" block into an if statement's condition slot, the blocks will not snap together. This simplifies and focuses the learning of computational thinking concepts by eliminating the time that one would spend chasing small syntax errors in traditional programming environments.

Scratch is designed to facilitate learning through tinkering, allowing students to create simple to complex projects while easily viewing the tweaks they are making to the program's logic [12]. The easy navigation and single window user interface simplify the testing and debugging process. Changing the logic of a program is less costly in the absence of syntax errors, thus they can make changes without the fear of "breaking" the rest of the code. Scratch also provides a large and interactive online community, where users can share their projects, adapt and "remix" other public projects, and communicate questions, compliments, or suggestions to other users. The community aspect is imperative in the development of computational thinking practices and perspectives. It is also offers students the opportunity to learn valuable social skills, related to programming and in general, by participating in and contributing to a community [14].

2.3 CS First

The free-form nature of Scratch is part of what makes it successful for encouraging novices to engage and experiment. However, when aiming to teach computational thinking to middle school students in a limited amount of time, it is beneficial to have a coherent and goal-oriented curriculum. Some teachers or club hosts may choose to design their own curriculum, but the COSMIC research effort leverages the set of free curricula provided by Google as part of their CS First computer science club initiative. CS First provides free materials along with numerous activities designed to attract students of varying interests [25]. These curricula and materials use Scratch as the programming language and environment for their activities. The curriculum is divided into clubs, each with a different theme. There are currently seven themes: “Art”, “Fashion and Design”, “Friends”, “Game Design”, “Music and Sound”, and “Sports.”. The clubs include 8 activities composed of an instructional video and Scratch project. Each activity focuses on a new CT concept and is designed to teach kids ages nine through fourteen computational thinking concepts and practices. The CS First curriculum is especially important and widely beneficial to the community in that it gives teachers, parents, and others who may feel they do not know enough about computer science or programming to teach their own computer science course.

Students also received a card stock paper “passport” at the beginning of each club, provided by CS First. Students used their passports to record their CS First/Scratch usernames, as well as to keep track of their progress for the theme. Stickers were also provided by CS First which were unique to each activity in the theme. Students received a sticker upon completion of the day’s activity to put it in their passport. When the club was finished meeting for the semester, students took their passports home as a souvenir from the club.

2.4 COSMIC Club Structure

This research was conducted and funded as a part of the "COSMIC: Computer Science for Middle Schools in Caldwell: A Pilot Approach to Targeting Underrepresented Middle School Students in Computer Science" grant from Google's non-profit Tides Foundation. This grant was awarded to Dr. James B. Fenwick, Jr. as the Principal Investigator, with Dr. Tracie Salinas as Co-Investigator; both Drs. Fenwick and Salinas are professors at Appalachian State University. Several after-school "COSMIC" clubs were conducted at multiple Caldwell County schools over a period of time from 2015-2017. COSMIC clubs were hosted by a teacher at the school, aided by Appalachian State University computer science graduate research assistants, and supervised by Dr. Fenwick. Hosting teachers had varying levels of prior computer science knowledge and experience, ranging from almost none to familiarity only with Scratch to having taught computer science related courses previously. Graduate research assistants were placed at each club meeting to aid the teacher with the material and answer students' questions, as well as to help students receive more one-on-one help and interaction by decreasing the student to teacher ratio. The level of involvement of hosting teachers ranged from completely "hands-off" to very involved.

COSMIC clubs met once a week for eight to ten weeks after-school for a period of time ranging from 45 to 75 minutes. Ideally, each student in COSMIC completed a CS First themed club (i.e. the "Sports" theme) over this period of time. COSMIC met eight times each iteration (around three months in a semester) with one to two additional meetings to account for absences or additional time needed. Due to the age of the children (11-14) and the nature of an after-school club, students were not made to strictly adhere to this ideal schedule of completion. How strictly students adhered

to the curriculum was ultimately up to the teacher who hosted the club. There were also several field trips where Caldwell County students would travel to Appalachian State University campus for STEM related activities or grant investigators traveled to Caldwell County for independent STEM activities.

COSMIC clubs ranged in student participation from as few as six students to as many as 25 students per club. Scheduling conflicts with other after-school activities or transportation issues accounted for inconsistent attendance or failure to complete a full COSMIC/CS First club. The research focused specifically on COSMIC clubs hosted at Colletsville, Gamewell, William Lenoir, and Granite Falls Middle Schools during the spring semester of 2017, as well as COSMIC clubs hosted at William Lenoir and Granite Falls Middle Schools during the fall semester of 2017.

Chapter 3 - Related Work

Numerous studies have been conducted with the aim of proving computational thinking (CT) is something that can be taught, understood, and quantified. While the context and methodology may vary across studies, the results are always positive: kids gain a better understanding of CT after participating in a club, class, or after-school program with computational thinking as its focus. This research highlights and builds upon studies that used Scratch as the tool to facilitate the learning of computational thinking.

Many studies have been conducted using Scratch as the primary or only method of teaching kids computational thinking. Brennan and Resnick put forth their definition of computational thinking and methods of assessment, where gauging computational thinking concepts stemmed mainly from analyzing Scratch projects for the kind and number of Scratch blocks used in the project [3]. For their sample data, they randomly chose or recruited Scratch users aged 8-17. They found that not only could computational thinking concepts be quantified through project analysis, but the other CT categories of practices and perspectives could also be quantified to give a richer and more multifaceted view of how kids learn computational thinking.

Other studies employed multiple methods to assess computational learning as well. One study used a combination of field notes, project analysis, and the Bath County Computer Attitude Survey (BCCAS) [4]. The BCCAS is used to gauge how kids K-12 feel about technology [1]. The survey presents many subjective statements, with three possible answers: “I don’t know,” “Agree” and “Disagree.” The BCCAS has been statistically validated as an assessment of computer attitude over time. In this study, Brown et al. sought to show the positive effect of computational thinking on the problem-solving skills of middle school students. They did this by incorporating a 45-minute

daily Scratch lesson into their standard educational curriculum for a month. At the end of the study, students who participated in the daily Scratch lessons showed a 9% increase in score between a pre- and post-test designed to elucidate problem solving methodology students use in solving mathematical questions. The control groups showed a 26% decrease between pre-and post-tests. This is a clear demonstration of the far-reaching effects of learning computational thinking, showing that it can be beneficial for cross-disciplinary problem solving.

Another study sought to find out what would happen if students learned programming concepts completely at their own discretion [11]. The context was a community technology center in an area of Los Angeles with low socioeconomic status, where kids ages eight through eighteen could come and use Scratch when they chose and for however long they liked. Aside from one occasion, the kids were not aided in their learning by computer scientists, but rather by graduate students who were put in the club to model learning. The graduate students had negligible prior knowledge and experience with programming and were new to Scratch.

Overall, 536 projects were collected from 80 students. Half of the 80 students were female and all were people of color [11]. The results of this research indicated that 425 out of 536 projects made use of scripts with sequential execution, and 88% demonstrated parallelism with multiple scripts running at the same time. When interviewed, students most often compared Scratch to paper or a sketchbook, because "it can be your own creative world." This is indicative of a perspective shift toward an active and creative relationship with technology, showing computational thinking perspective. Interestingly, students did not, for the most part, equate Scratch scripting with "computer programming." the authors postulate this actually helped facilitate learning Scratch because it did not adhere to any stigma or preconceived notions about programming, but rather was seen as something "cool."

Without instructor intervention, students discovered and implemented the computational thinking concepts of loops, conditionals, communication, synchronization, and less commonly, more complex concepts such as variables and Boolean logic. This study shows that kids can learn computational thinking concepts and perspective purely through access to a computer with Scratch and an intrinsic motivation to learn. Thus, it young students may benefit from the aid of computer science students, but it is not critical that a computer science club or course be led by someone with extensive computer science knowledge.

Chapter 4 - Data Collection Methodology and Tools

To gauge the success of COSMIC, data was collected via several avenues: CS First's website tracked students' completion of activities using links to their shared Scratch projects, qualitative data from surveys conducted by COSMIC club research assistants was obtained, and Scratch projects themselves were quantitatively examined and analyzed.

4.1 CS First Data

CS First themed clubs are comprised of 8 activities designed to take approximately one hour. These activities are designed to introduce students to programming concepts such as design, logic, events, and testing. Each activity begins with the student watching a short video introducing them to the concept and objective of the lesson before they are directed to the Scratch programming interface where they implement the objective of the lesson. Activities are structured so that students develop a project in steps, having several videos and small objectives that build upon one another, aligning with the practice of being incremental and iterative set forth in our definition of CT [3].

The project aspect of CS First is open to interpretation by students and does not have a "grading" system in place to ensure that they have achieved the objective of the lesson. The success of the student in achieving the lesson's objective is up to the discretion of the student and teacher who is "hosting" the CS First club. Club hosts are given lesson plans by CS First that include instructions, an overview of the activity, example projects, as well as suggestions for each activity such as "Look for students who are designing (adding sprites, backdrops, etc.), and encourage them to experiment with code" [26].

Along with lesson plans, hosts are provided a link to an example completed project for each activity that meets all of the criteria from the activity videos. The sample solution project also

includes notes to the instructor with a summary/outline of the videos and instructions like “To run, click the green flag.” Figure 4.1 contains a sample solution Scratch project for the first activity of the Storytelling theme: “Dialogue.” This sample solution is a project that meets the criteria for this activity’s objective; In this case it contains one script with instructions for two characters “speaking to” one another when the green flag is clicked. A “solution sheet” is also included in the materials, and these provide instructors with a step by step solution to each activity in a theme, showing images of what the blocks should look like at each step [26]. Figure 4.2 shows a detailed look at the solution sheet steps for Activity 1 “Video 3: Speaking and Responding”, part of the “Storytelling” theme.

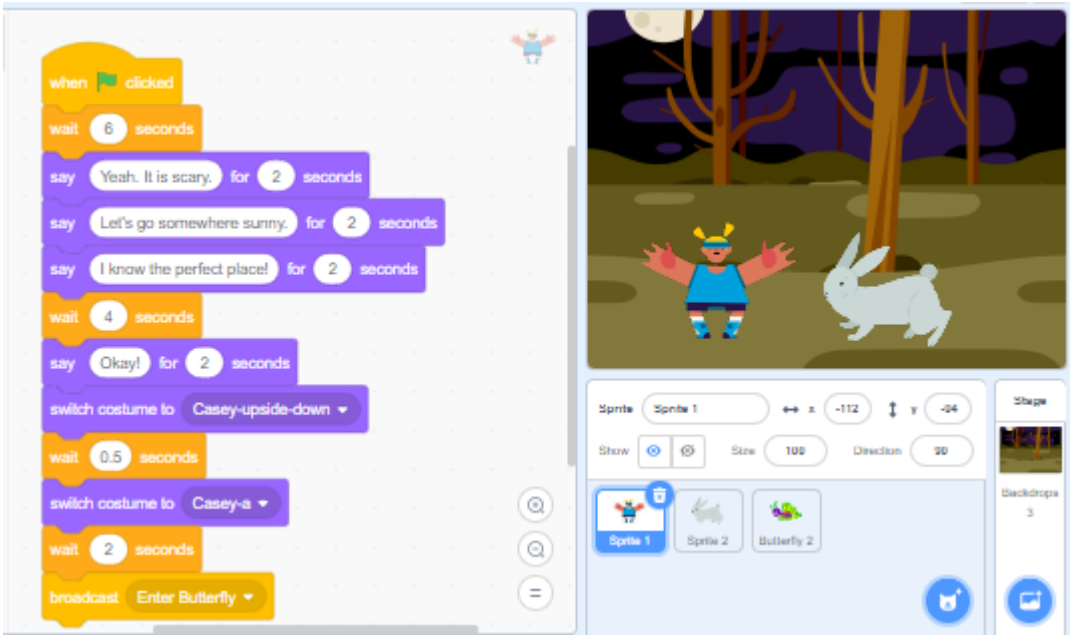


Figure 4.1 Sample Solution "Dialogue Example Project."

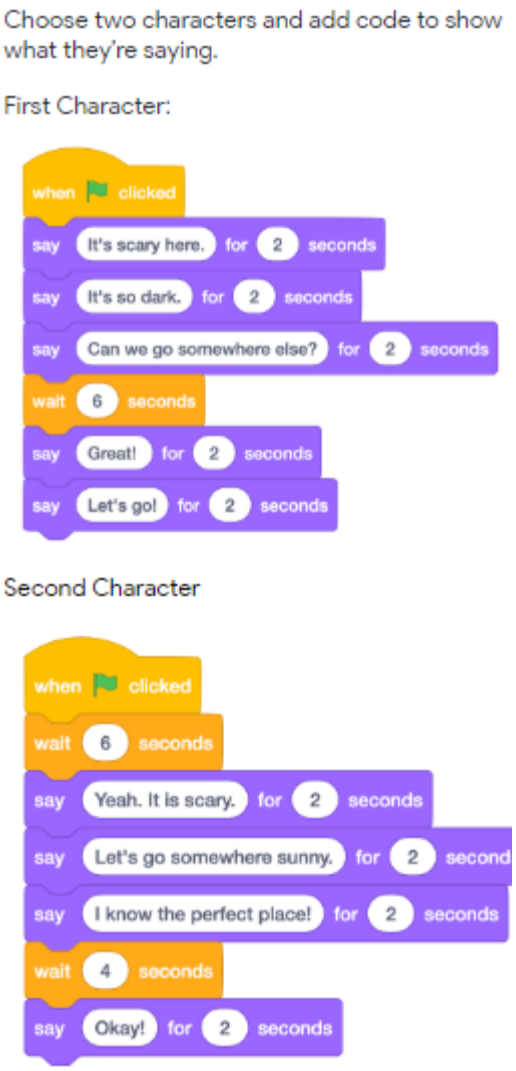


Figure 4.2 Example Solution Sheet.

CS First tracks a student's "module completion" by indicating in a table accessible to the host whether the student has watched at least one video for the activity or watched all the videos for each activity of that clubs "theme." Scratch projects created by the student for that theme are also accessible if the student chooses to share them with the host. Projects are not automatically shared

with the host of the club but are saved under the student's individual Scratch account user name. Some students were assigned Scratch user names by the host of the club for use during club meetings while other students used their already active personal accounts to complete projects for the club.

The CS First curriculum includes pre and post surveys as well as a short "Reflection" at the conclusion of each activity that may be completed by the student. The reflection portion of the activity asks students how they felt about the day's activity on a 7 point scale from "It was the worst club ever!" to "It was the best club ever!" The reflection also contained several varying questions that asked about their opinion on computer science, and how they used computer science to solve a problem. These surveys intended to capture students' feelings toward computer science, their feelings about each meeting's activity, as well as assessed their conceptual knowledge using Scratch "coding" questions.

4.2 Qualitative Data Collection

An important aspect in the Brennan and Resnick framework for CT is perspective. While qualitative analysis is less defined and more subjective than quantitative analysis, it can also allow a more nuanced view into how students learn and how they feel and think about what they're learning. In order to completely articulate a framework for viewing computational thinking, one must account for the shift in perspective that occurs when a student's relationship to the technological world around them changes through learning [3].

4.2.1 Surveys

In addition to the surveys included in CS First, students were given an anonymous post survey created by COSMIC researchers at the conclusion of the spring 2017 session to collect demographic data as well as information about how they felt toward the club and its materials, their

feelings toward computer science, and how they first learned about writing code. In the fall of 2017, expanded pre and post surveys were administered at the beginning and end respectively of the fall 2017 club through Google Forms. These supplementary surveys were intended to paint a clearer picture of each student's relationship with technology, their background, and their comprehension in order to gauge the efficacy of COSMIC club in teaching computer science concepts and fostering a positive relationship between students and the technological world around them. Survey data serves as a supplement to the field experience and the notes of graduate research assistants who attended the COSMIC club meetings.

4.2.2 Interviews

At the conclusion of COSMIC in the fall 2017, five students who participated in the Granite Falls iteration of COSMIC as well as five students at William Lenoir were interviewed by the supervising research assistant. Permission was given by the parent or guardian to record audio of these interviews for the purpose of gaining insight regarding students' background, experience in COSMIC, and future academic plans. These verbal interviews were either conducted individually or in groups of two to three. Group interviews tended to promote a more casual conversational interview, while individual interviews were more formal and required more prompting from the researcher. Interviews have the same basic goals but changed and evolved to fit the individual student and flow of the interview. Interviews were not conducted in a predetermined amount of time but instead were limited either by the time allotted for the club meeting's activities or ended naturally when the interviewer felt the student was ready to be done with the interview. The overarching goal of these verbal interviews was to elicit the more subtle aspects of computational thinking perspective. A student's perspective and relationship with computer science and technology

in general is multifaceted and nuanced, making it harder to elucidate than conceptual knowledge. The researcher sought to gain insight into students' individual backgrounds and feelings about their past and future computer science experiences.

4.3 Quantitative Scratch Project Analysis

The complexity of a Scratch project is multi-faceted and without tools requires knowledge of computer science concepts to quantify. In this section the structure of a Scratch project is examined as well as tools that automate the quantifying of Scratch project completeness and complexity in terms of standard computer science and computational thinking concepts. A number of tools are briefly described; however, this research uses the Dr. Scratch tool so it has a longer discussion.

4.3.1 Scratch Project Structure

Scratch projects can be downloaded from the Scratch website, provided the owner has shared the project. Projects are downloaded as a .sb file, depending on when the project was created. A .sb file can be treated as a zip file and opened and extracted as any other zip file. An .sb file contains images used in the project, sounds used in the project, SVG (Scalable Vector Graphic) files describing "sprites" used in the project, as well as a project.json file describing the structure of the project [24]. The project.json file includes a break down of the project into individual stages, costumes, sprites, sounds, scripts, blocks and their associated events and/or functions, all recorded in the JavaScript Object Notation (JSON) format. Thus one way to analyze the complexity of a Scratch project is to examine this file and count the various components that make up the Scratch project. There are several tools available that perform this type of quantitative analysis of a Scratch project file.

4.3.2 Hairball

While approaches of Scratch project analysis for computational thinking have been proposed, they are mainly based on manual analysis [15]. Manual analysis may appear simpler in that it may not require actually downloading the breakdown of the Scratch project, it would also require the person doing the analyzing to thoroughly understand the programming language as well as the concepts that are intended to be demonstrated in the project. Being that this resource (individuals qualified and willing to do the manual analysis) is limited, researchers have sought to develop tools which would automate this process [15].

In 2013, a tool called Hairball was proposed by researchers at the Special Interest Group on Computer Science Education (SIGSE) conference [2]. Hairball is an automated system that can be used by students and teachers to highlight potential issues in Scratch programs such as unused code (code that never gets executed), errors in synchronization between interactive blocks, and "unsafe practices" [15]. Hairball was intended for use in conjunction with manual scripts and requires a basic understanding of command line execution of Python scripts to use which is potentially beyond the understanding of many students and teachers in our target audience [15]. However, the plug-in based architecture of the Hairball provided a platform on which another automated tool, Dr. Scratch, was built.

4.3.3 Dr. Scratch

Dr. Scratch allows users to analyze a scratch project for computational thinking concepts by either uploading a .sb1 or .sb2 file, or simply providing the URL of the scratch project [27]. Dr. Scratch then provides feedback on the project with respect to demonstrated Computational Thinking concepts such as "Flow Control", "Data Representation", and "Abstraction". The project is given an

overall CT score between 0 and 21 and assign a level: “basic, “developing”, and “proficient.” Computational concepts are defined by Dr. Scratch and evaluated in the following ways:

- Flow Control: Instructions that help you control the behavior of your characters. For example, an if statement that makes several blocks repeat until a condition is met [27]. A program with the most basic flow control would be one with a series of blocks that execute once in order, whereas a program with more advanced flow control might repeat a series of blocks for a certain amount of time. Advanced flow control would have a series of blocks repeat until a condition is met, i.e. move left until you touch another character.
- Data Representation: How data is used and stored in the Scratch program. Every sprite in Scratch has a number of attributes that can be used or modified within the program: position of the character on the stage, size of the sprite, orientation or direction the sprite is facing, the costume the sprite is wearing, and the visibility of the sprite (whether it is shown or hidden). A basic level of data representation is judged as using and modifying these default attributes. More advanced data representation would be if the user creates and/or modifies their own variable i.e. to hold an integer value. The most advanced level of data representation is considered to be the use of lists, which are manipulated like arrays where the user can insert, delete, or choose from a list of values they create.
- Abstraction: Projects that are demonstrative of the ability to break a large problem down into smaller more consumable parts receive a higher Abstraction score. A project with a low abstraction score would perhaps have every operation included in the program all part of a single script with some repeating code for operations that happen multiple times. A program with a higher abstraction score might have a seperate user created function that is called twice from the “main” script, instead of the code that needs to be repeated being written out

twice. The highest abstraction concept according to Dr. Scratch is the "cloning" function, which allows users to instantiate multiple objects with the same behavior, i.e. instantiates multiple objects from the object "definition", similar to a class [27].

- User Interactivity: Scratch provides blocks and functionality to allow users to control and/or affect the program. The most basic user interaction is the "When [green flag] clicked" block. This is generally the entry point for most Scratch programs and begins the "main" script when the user clicks the green flag. Scratch programs can be designed to ask users questions, or have them move a sprite, or at the most advanced level to use their microphone or webcam to interact with the program.
- Synchronization: Synchronization in this instance refers to the behavior of sprites in a Scratch program occuring in the order we want. For instance, when programming two characters who are interacting, one might use the wait blocks to make their "conversation" occur in order. A more advanced way of synchronizing the conversation between two characters would be to use the broadcast/receive blocks. In addition, blocks can be used to make things happen when another event occurs, i.e. a variable is set to a certain value or when the backdrop changes.
- Parallelism: Dr. Scratch defines parallelism as the "possibility that several things occur simultaneously" [27]. Users can have several scripts that all execute simultaneously when the green flag is clicked. Scripts can also be triggered simultaneously by pressing a key or clicking a sprite. Events like the backdrop changing, receiving a message, webcam or microphone interaction, etc. can also trigger simultaneous actions.
- Logic: Dr. Scratch defines logic as dynamic instructions that behave differently depending on the situation. Logic scores range from the most basic (a linear story, maybe the use of an

if statement), to the use of logical operations to check for multiple conditions at once (use of the AND, OR, if/else, and No operations).

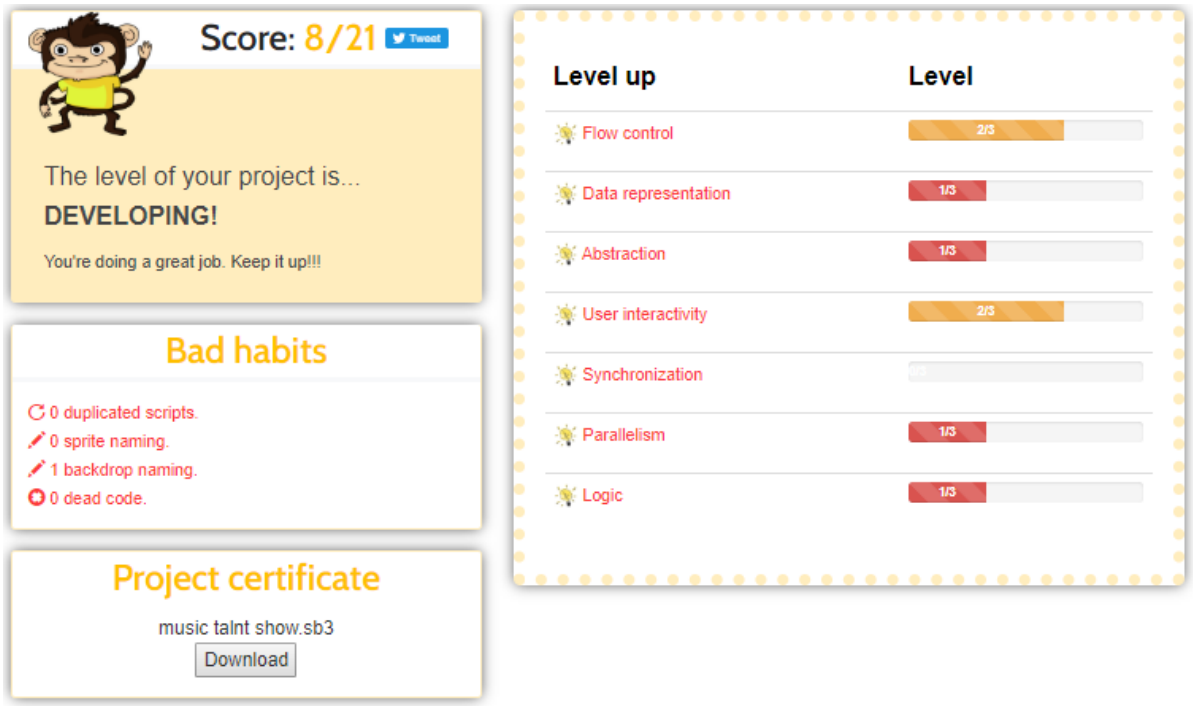


Figure 4.3 Dr. Scratch analysis.

Figure 4.3 shows an example of Dr. Scratch project analysis results for a project called "music talnt show" [sic], with concept scores are shown on the right, and the overall score and best practices shown on the left. Dr. Scratch also provides feedback regarding "best practices" such as naming conventions used (Did they leave the default names like "Sprite1", "Sprite2" or did they change the names to make the program easier to read?), duplicated scripts, attribute initialization, and unexecuted code snippets. These best practices do not factor into the overall CT score of the project but are noted in their own section of the score page.

This quantitative CT feedback aligns with the two facets of CT mentioned earlier in our working definition of CT from Brennan and Resnick's paper: CT concepts and best practices [3]. Dr. Scratch also links information that Scratch programmers (Scratchers) can use to improve on their own.

Dr. Scratch provides a comprehensive definition of the criteria by which they designate a "level" for the project. Table 4.1 shows the breakdown of CT concepts scores provided in the paper "Dr. Scratch: Automatic Analysis of Scratch Projects to Assess and Foster Computational Thinking" [14].

Table 4.1 Dr. Scratch rubric.

CT Concept	**Competence Level**			
	Null (0)	**Basic (1 point)**	**Developing (2 points)**	**Proficiency (3 points)**
Abstraction and problem decomposition	-	More than one script and more than one sprite	Definition of blocks	Use of clones
Parallelism	-	Two scripts on green flag	Two scripts on key pressed, two scripts on sprite clicked on the same sprite	Two scripts on when I receive message, create clone, two scripts when %s is > %s, two scripts on when backdrop change to
Logical thinking	-	If	If else	Logic operations
Synchronization	-	Wait	Broadcast, when I receive message, stop all, stop program, stop programs sprite	Wait until, when backdrop change to, broadcast and wait
Flow control	-	Sequence of blocks	Repeat, forever	Repeat until
User Interactivity	-	Green flag	Key pressed, sprite clicked, ask and wait, mouse blocks	When %s is %s, video, audio
Data representation	-	Modifiers of sprites properties	Operations on variables	Operations on lists

4.3.4 Other Tools

Lacking access to source code for the Hairball and Dr. Scratch projects restricts researchers from exploring enhancements to these tools. Thus, at Appalachian State University, Dr. Fenwick

directed several student research efforts to build extensible tools that perform similar analysis. These tools are useful and part of continuing research efforts, but are not yet as robust as Dr. Scratch. SCATT, a tool for Scratch project analysis, prints a report of a project's sprite usage including a breakdown of how many scripts the sprite was used in, how many costumes the sprite had, what blocks were associated with the sprite, and what variables and lists were associated with the sprite. SCATT also prints a total count of sprites, scripts, variables, lists, and blocks of each type used in the project. A report of this nature can be useful for an instructor to determine the complexity of a project, but still requires a certain level of knowledge of Scratch and CT concepts to distinguish quantity from quality, i.e. a lot of things thrown into a Scratch project haphazardly producing a high count in the report versus a lower count of components that work together to better demonstrate CT.

Another tool, SPAE, was built on top of SCATT to analyze Scratch projects through a web accessible interface given the project's ID or URL. SPAE checks each submitted project ID to ensure the project exists in Scratch and is shared, and then checks to see if the project is a remix of any previous Scratch projects. SPAE then provides an aggregate report similar to SCATT with various counts for blocks, sprites, and variables. While an improvement upon SCATT, SPAE's output has similar limitations still requires CT knowledge to accurately interpret the quality of the project in demonstrating CT concepts.

4.4 Data Analysis Methodology

More than 210 Scratch projects were collected for analysis from students throughout all COSMIC club sessions. More than 60 students were surveyed about their COSMIC experience and relationship with technology and Computer Science (CS), and 10 students participated in recorded interviews with a research assistant who participated and helped supervise multiple iterations of

COSMIC club. The field experience of research assistants participating in COSMIC club was also considered.

4.4.1 Dr. Scratch Analysis

In 2017, a comprehensive analysis was performed on student data obtained from COSMIC club projects using the Dr. Scratch tool for project analysis. To establish a baseline with which to compare COSMIC participant projects, the "example" completed projects for each CS First theme's activities were also analyzed through Dr. Scratch.

Table 4.2 A breakdown of the Dr. Scratch analysis for the "Game Design" theme's activity example projects.

Game Design	Flow control	Data representation	Abstraction	User interactivity	Synchronization	Parallelism	Logic	Total	
Activity 1	1	1	1	1	0	1	0	5	Basic
Activity 2	3	1	1	2	0	0	3	10	Developing
Activity 3	2	1	0	2	2	0	1	8	Developing
Activity 4	3	1	1	2	3	0	3	13	Developing
Activity 5	2	1	1	2	0	1	3	10	Developing
Activity 6	3	2	3	2	0	1	3	14	Developing
Activity 7	2	1	1	2	3	3	1	13	Developing
Activity 8	2	2	1	2	2	1	2	12	Developing
Average	2.25	1.25	1.125	1.875	1.25	0.875	2	12.14285714	Developing
Max	3	2	3	2	3	3	3	14	Developing

As seen in Table 4.2, the example solution projects provided by CS First as part of their curriculum ranged in complexity levels according to Dr. Scratch analysis. It was therefore important to analyze the solution projects to establish the target complexity for a student completed project. It is also important to note that some activities included "starter" projects for the students to build upon. To account for this, the starter projects for each activity, when applicable, were also run through Dr. Scratch to attain a complexity score.

Table 4.3 The Dr. Scratch breakdown of the provided starter projects for activities in the "Game Design" theme.

Activity	Flow Control	Data representation	Abstraction	User Interactivity	Synchronization	Parallelism	Logic	Total	
Cave Surfer	2	1	1	1	0	1	0	6	Basic
Gaming	-	-	-	-	-	-	-	-	-
Launcher	0	0	0	0	0	0	0	0	Basic
Platformer	0	0	0	0	0	0	0	0	Basic
QuestGame	2	1	0	2	1	0	1	7	Basic
Racing Game	0	0	0	0	0	0	0	0	Basic
Sky Maze	0	0	0	0	0	0	0	0	Basic
Average	0.667	0.333	0.167	0.5	0.167	0.167	0.167	2.167	Basic
Max	2	1	1	2	1	1	1	7	Basic

Table 4.3 displays the score of the starter projects. These scores are needed to ascertain the difference between the starter project's complexity score and the student's project's complexity score. Particularly in cases where the starter project already has a "Developing" score for an activity, a final project submitted by a student could have a "Developing" complexity score without the student actually adding anything to the starter project. A student project built on either no starter project or a starter project with a "Basic" low complexity score that attains a "Developing" complexity score would demonstrate much more student work and understanding.

Only data from "engaged students" was considered, with engaged students being defined as students who completed at least 5 out of the 8 activities in a theme, had projects that were shared and accessible, and attended either the last or second to last club session. This is intended to ensure that the data set collected was complete enough to draw meaningful conclusions. Students being required to attend the second and/or last session ensured that students completed at least one of the more complex activities and not just the beginning/easier activities. This definition of engaged students protects the integrity of the data and conclusions that may be drawn from the data regarding progress in CT. Of the 94 distinct students from which data was collected, 37 met the definition of an "engaged student." This is due to students working on other curriculum (CodeHS or SLNova), projects not being shared by students, as well as the every day challenges of after-school club

engagement where students also participated in other after-school activities like sports, band, and other clubs that sometimes interfered with COSMIC attendance.

Each student was assigned an id of the format <cs######> where the #'s represent a six digit number. These ids were unique to the student and were used to keep track of their CS First and Scratch work (the id was used as their login username for CS First and Scratch, unless the student elected to use their personal or pre-existing Scratch account in which case they would not be considered in this data set).

4.4.2 Sentiment Analysis

Sentiment analysis, or opinion mining, is a field of study that analyzes people's sentiment or general feelings toward entities such as products, services, individuals, issues, events, and topics [21]. With the ubiquitous nature of social media today, sentiment analysis has become a major topic of research with a wealth of data available. This kind of natural language processing is geared toward generally gauging whether a person feels positively or negatively toward the entity being discussed [21].

Python is an open-source programming language that can be used by anyone to build programs that process and analyze text data [22]. The Natural Language Toolkit (NLTK) is a platform for building Python programs that analyze human language data [23]. This research uses TextBlob, which is built on top of the NLTK, to perform sentiment analysis on textual data collected through interviews of COSMIC participants. TextBlob is an Application Programming Interface (API) that allows programmers to perform natural language processing tasks such as part-of-speech tagging, noun phrase extraction, sentiment analysis, classification, and more [24].

The sentiment analysis feature of TextBlob takes as input a body of text and produces a polarity score. The polarity score produced for input text data falls in the range [-1, 1], where -1 is the most negative sentiment and 1 is the most positive [24]. A polarity score is produced using a large corpora of text with values assigned to words based on the sentiment of the word (how positive, negative, or neutral it is). More objective words are assigned a neutral polarity score, where words expressing or indicating emotions like “happy,” “sad,” “angry,” “excited,” are more subjective and are assigned a positive or negative polarity score.

As part of this research, a Python program was developed that performed a TextBlob analysis of the COSMIC interview data. The TextBlob’s sentiment analysis feature produced a polarity score for interviews given by COSMIC participants. Transcripts of interviews were modified only to remove interviewer’s questions and speech in order to not affect polarity score. Only the interview subject(s) responses were used as input for the sentiment analysis program.

Chapter 5 - Results

The previous chapter described the sources and types of quantitative and qualitative data collected. This chapter discusses the specifics of the collected data and draws some conclusions on the efficacy and success of COSMIC efforts to promote and instill computational thinking concepts, practices, and perspectives in Caldwell County middle schools. This section also discusses COSMIC efforts with regard to teacher education and participation overall. This section summarizes the first two years of COSMIC through information obtained from internal grant evaluations. The results of quantitative project analysis and qualitative survey and interview analysis are the primary aspects discussed.

5.1 Internal Evaluation Reports

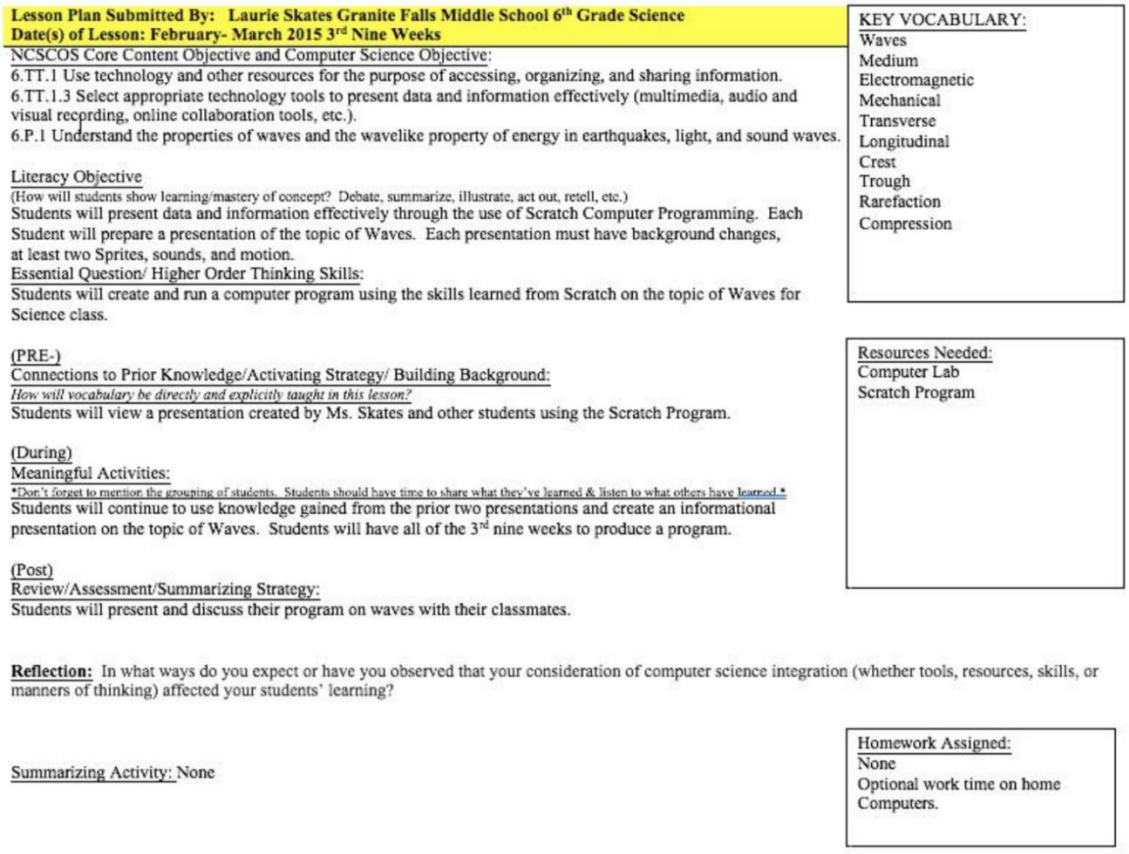
As part of internal grant reporting requirements, the COSMIC grant investigators submitted interim evaluation reports[19]. For the sake of completeness, this thesis presents a summary of those reports in this subsection.

5.1.1 COSMIC Year 1

The first year of the COSMIC grant work comprised two academic terms (Spring 2015, Fall 2015), one summer camp for students (Summer 2015), and one summer camp for teacher professional development (Summer 2015). More than 80% of 153 unique students who participated in COSMIC demonstrated increased positive perceptions of computer science as measured by pre-surveys, field notes, and artifacts from club and camp meetings. 59% of students demonstrated proficiency in 4 or more competencies as measured within the CS First System. 47% of participating students were female and 17% were part of a minority group.

Student journals from summer camps are reflective of increased confidence in computer science: “Seventeen completed journals were collected, and all reported increased understanding of computer science and confidence in computer science skills. When prompted regarding what they might like their families or teachers to know, students responded with statements of ‘I can…’ or ‘I did…’ or ‘I made…’, all strong sentences referring to specific skills or accomplishments.” Student journals reflect a deeper and more nuanced understanding of computer science over time, with 13 out of 17 collected journals showing a shift toward viewing computer science as a method of problem solving. These journal entries also showed students had an increased perception and understanding of what a computer science career entails as well as what sort of options are available in terms of a future career in computer science.

Teachers also received more exposure to and an increased understanding of computer science, which facilitates more student exposure to computer science career options through field trips, career fairs, and knowledge transfer. During the fall 2015 term, 7 teachers submitted a total of 21 lesson plans that incorporate Scratch programming activities into classroom subject courses ranging from math to language arts to science. In figure 5.1, we see a lesson plan submitted by a Granite Falls Middle School (GFMS) 6th grade science teacher which incorporates Scratch as a medium for students to create and present a project on the properties of waves.



Lesson Plan Submitted By: Laurie Skates Granite Falls Middle School 6th Grade Science
Date(s) of Lesson: February- March 2015 3rd Nine Weeks

NCSCOS Core Content Objective and Computer Science Objective:
6.TT.1 Use technology and other resources for the purpose of accessing, organizing, and sharing information.
6.TT.1.3 Select appropriate technology tools to present data and information effectively (multimedia, audio and visual recprding, online collaboration tools, etc.).
6.P.1 Understand the properties of waves and the wavelike property of energy in earthquakes, light, and sound waves.

Literacy Objective
(How will students show learning/mastery of concept? Debate, summarize, illustrate, act out, retell, etc.)
Students will present data and information effectively through the use of Scratch Computer Programming. Each Student will prepare a presentation of the topic of Waves. Each presentation must have background changes, at least two Sprites, sounds, and motion.
Essential Question/ Higher Order Thinking Skills:
Students will create and run a computer program using the skills learned from Scratch on the topic of Waves for Science class.

KEY VOCABULARY:
Waves
Medium
Electromagnetic
Mechanical
Transverse
Longitudinal
Crest
Trough
Rarefaction
Compression

(PRE-)
Connections to Prior Knowledge/Activating Strategy/ Building Background:
How will vocabulary be directly and explicitly taught in this lesson?
Students will view a presentation created by Ms. Skates and other students using the Scratch Program.

Resources Needed:
Computer Lab
Scratch Program

(During)
Meaningful Activities:
Don't forget to mention the grouping of students. Students should have time to share what they've learned & listen to what others have learned.
Students will continue to use knowledge gained from the prior two presentations and create an informational presentation on the topic of Waves. Students will have all of the 3rd nine weeks to produce a program.

(Post)
Review/Assessment/Summarizing Strategy:
Students will present and discuss their program on waves with their classmates.

Reflection: In what ways do you expect or have you observed that your consideration of computer science integration (whether tools, resources, skills, or manners of thinking) affected your students' learning?

Summarizing Activity: None

Homework Assigned:
None
Optional work time on home Computers.

Figure 5.1 Lesson plan submitted by a Granite Falls Middle School 6th grade science teacher.

5.1.2 COSMIC Year 2

The second year of COSMIC contained two academic terms (Spring 2016, Fall 2016), with separate summer camps for teachers and students held in the summer of 2016 [20]. This subsection presents a summary of the grant investigators internal evaluation report for year 2.

In 2016, COSMIC continued at four Caldwell county schools. After school club meetings saw the participation of 84 students using the StarLogo Nova curriculum. COSMIC camp was held in the summer of 2016 beginning the usage of CS First curriculum in conjunction with Scratch. A total of 20 teachers, 13 of which are female, received professional development and training to

integrate Computational Thinking (CT) and computer science concepts into their regular course curriculum. In the fall of 2016, after school club meetings also began using CS First and Scratch curriculum, with 113 students participating.

Half of the students in after-school clubs and summer camps were female and 15% were underrepresented minorities. One student wrote about her experience in COSMIC summer camp:

> *"Cosmic summer camp was a very fun experience ... I believe the industry needs more females in it and this camp was a way for young girls who are just figuring out who they are to get involved and learn while having fun. I didn't even realize that I was learning until I came home and looked back at everything I did.*

Professional development for teachers increases the likelihood that they will be willing to incorporate CS and CT concepts into their lesson plans for their regular classes. Instilling confidence in teachers that they can host a CS club like COSMIC even if they are not familiar or confident in their CS knowledge is part of an effort to increase exposure of students to CS/CT at an earlier age. At the end of 2016, three teachers agreed to take on mentoring roles to help teachers integrate computer science (CS)/computational thinking (CT) concepts into their regular curriculum. 10 out of the 20 teachers who participated in the summer 2016 sessions indicated that they will implement CS and CT into their regular curriculum.

5.2 Quantitative Analysis

A total of 210 student projects from 37 engaged students participating in the COSMIC clubs held in 2015 and 2016 were analyzed by the Dr. Scratch project analysis tool. The scores for these projects were then aggregated and analyzed by research assistants at Appalachian State University. Each student's project for each activity of a theme was run through Dr. Scratch to produce their CT

score and the point breakdown by CT concept as discussed in section 4.4.3. It was noted for each project whether or not the student used a starter project, which starter project they used (often several are provided for an activity to allow for student personalization), and what the CT score of their starter project was for comparison to the project's final CT score. Scores for each CT concept were then summarized, and the percentage of students scoring 0-3 respectively were calculated.

Table 5.1 The summary data breakdown for CT concept scores for all projects analyzed from 2017.

	Flow Control	Data Representation	Abstraction	User Interactivity	Synchronization	Parallelism	Logic
3	23.47%	0.94%	12.68%	0.00%	8.92%	10.33%	26.76%
2	52.58%	23.94%	4.23%	69.01%	19.25%	6.57%	8.92%
1	11.74%	57.28%	61.97%	18.31%	19.72%	43.66%	22.07%
0	2.35%	7.98%	11.27%	2.82%	42.25%	29.58%	32.39%

As seen in Table 5.1, the CT concepts with the highest percentage of students scoring a 3 on average were Logic with 26.76% of students scoring a 3, followed by Flow Control with 23.47%. The concept with the highest percentage of students scoring 0 was Synchronization with 42.25%. Investigating this low scoring Synchronization result deeper revealed that the CS First curriculum themes do not make heavy use of the Synchronization concept as measured by Dr. Scratch. Indeed, the average Synchronization score for CS First solution projects is only 1.23. Thus, the COSMIC student performance in this concept column should not actually be considered poor, but rather on par for the CS First curriculum expectations.

Student data was collected for every theme, with the smallest number of viable projects in a theme across schools being 5 in the Fashion & Design theme. The most chosen and worked on theme was Game Design with 134 student projects from 3 schools. The second most chosen theme

was Art with 28 viable projects. Schools sometimes choose for all students to participate in a single theme (Granite Falls Middle School and Gamewell Middle School), while others narrowed themes down to those with the most interest and let students choose from a couple (Colletsville Elementary School, William Lenoir Middle School). Notably the school with the most students, Granite Falls Middle School, had all of its students participating in the Game Design theme. Most likely, the Granite Falls teacher wanted all students to be working on the same theme. A full breakdown of projects by school can be seen in table 5.2.

Table 5.2 A breakdown of the number of projects analyzed for each theme by school.

School	Theme	Projects
CES		
	Animation	11
	Art	0
	Social Media	0
	Game Design	9
	Fashion & Desig	5
	Music & Sound	21
School Total:		46
GMS		
	Animation	0
	Art	0
	Social Media	0
	Game Design	5
	Fashion & Desig	0
	Music & Sound	0
School Total:		5
GFMS		
	Animation	0
	Art	0
	Social Media	0
	Game Design	120
	Fashion & Desig	0
	Music & Sound	0
School Total:		120
WLMS		
	Animation	0
	Art	28
	Social Media	11
	Game Design	0
	Fashion & Desig	0
	Music & Sound	0
School Total:		39
Total		210

Table 5.3 Student and starter project average scores.

Theme	Starter	Student	Difference
Animation	3.14	4.27	1.13
Art	0.63	7.68	7.05
Social Media	0.00	7.09	7.09
Game Design (w/ GFMS)	1.86	9.65	7.79
Game Design (w/o GFMS)	1.86	5.43	3.57
Fashion & Design	3.00	4.80	1.80
Music & Sound	2.33	7.62	5.29

Table 5.3 shows the breakdown of the average score of the starter projects and the student projects for each theme along with the difference. Game Design was separated into two sets - one for students at Granite Falls Middle School (GFMS) group which made up the vast majority, and one for students not at GFMS. The theme with the highest scoring starter projects (project supplied with the activity that the students are allowed to build upon) on average was the animation theme. This is due to the starter project providing significant project behavior initially. The animation theme was also the theme with the lowest average student score; meaning that student contributions were smaller. Average student scores were higher than the starter projects average score for all themes, thus on average work was accomplished by the students. The largest differential between average student scores occurred in the Game Design with Granite Falls Middle School group with 7.79 points difference, followed closely by Art and Social Media. This data suggests that some themes, like Art and Social Media, require more student effort.

Male and female students averaged similar scores, with males scoring a 9.80 project average, and females scoring an 8.71 project average. 92% of our sample group moved up a Dr. Scratch mastery level between the beginning of COSMIC and the end. Students who participated in the Art

and Fashion and Design themes had projects scoring on average higher than the solution projects, meaning that they regularly exceeded the target complexity for their projects.

5.3 Qualitative Analysis

This subsection will discuss the qualitative analysis of data from CS First and research assistant created surveys. The sentiment analysis of student interview transcriptions will also be presented.

5.3.1 Survey Data

On the pre-survey and post-survey administered as part of CS First curriculum, students were presented with the statement “If I get stuck on a computer science problem, I know how I might fix it," and asked to answer either “Strongly agree,” “Agree,” “Neutral,” “Disagree,” or “Strongly disagree.” The most common answer among male students on the pre-survey for this question was “Strongly agree”, while the most common answer for female students was “Agree.” A significant increase was seen in the percentage of both male and female students answering this question “Strongly agree” in the post-survey, as seen in table 5.4.

Table 5.4 "If I get stuck on a computer science problem, I know how I might fix it" survey results.

	Male		Female		Total	
	Pre-	Post-	Pre-	Post-	Pre-	Post-
%SA	22.22%	47.37%	17.65%	58.82%	19.44%	41.67%
%A	38.89%	31.58%	35.29%	41.18%	36.11%	36.11%
%N	33.33%	10.53%	35.29%	0.00%	36.11%	16.67%
%D	5.56%	5.26%	11.76%	0.00%	8.33%	5.56%
%SD	0.00%	5.26%	0.00%	0.00%	0.00%	2.78%

In the "Reflection" portion of CS First activities, students often use verbiage that demonstrates the desired shift in CT perspective. For example, in Storytelling Activity 6 when prompted with the short answer question "What would be the best part about being a computer scientist?", answers included:

- *"having fun and making stories/games on scratch"*
- *"Creating what you want"*
- *"making online books and games"*

This language is indicative of an active relationship with computation in which the student views computation as a medium with which to express themselves and create [3].

In Activity 7 of the same theme, when asked what they thought of the day's activity, answers included:

- *selected "I loved it!" option, with comment: "[I] loved to make my own story (even if it did have bugs)"*

- *selected "I liked it" option, with comment: "Don't give up instantly go back and read carefully"*
- *selected "I loved it" option, with comment: "To take your time and it is ok if you mess up."*

Once again words like "make" demonstrate an active relationship with computation. All three answers denote an understanding of CT practices. The first answer notes their program has bugs, demonstrating an awareness and understanding of the CT practice "Testing and debugging". The third answer also reflects an understanding of this CT practice by noting that it's okay if everything doesn't go according to plan. All three answers are representative of understanding the CT practice of "Being incremental and iterative" [3], which includes strategies like going back to improve upon your work.

In Activity 3 of the Game Design theme, students were asked the question "There's a new student at your school. What would you say to her to get her interested in computer science?" Answers included:

- *"[I] will say you will have fun"*
- *"Science is awesome. You really need to do it. It is awesome"*
- *"[It's] fun, you get to make games, you learn a lot, and get to be creative."*
- *"There is a lot you can do in computer science if you like video games than that is computer science."*

These answers express enthusiasm and regard of computer science as a versatile medium to create with and learn. This expression of science as a vessel for creativity embodies the perception of a computational thinker, as computation is a creative human activity [21].

In the survey given by research assistants to 23 participating students in the COSMIC spring of 2017, 74% said that they loved the COSMIC program. 78% said that they planned on taking a

computer science course in high school if it was an option. Of the 23 students, 13 of which were female, 91% responded that they had a better understanding of what a career in computer science would be like. In other reflections, students wrote of the COSMIC experience:

> *"It was an eye-opening experience to realize that Scratch could be used for things other than useless puzzles. It was a lot more interesting than when I was taught before."*

5.3.2 Interviews

Interviews gave insight into students' perception of computer science and the role it plays in their life both present and future. Students were asked about their introduction to computer science, their role models, their future plans, and their general feelings toward computer science. Group interviews were more enthusiastic, with students readily volunteering to be interviewed and often vying for the spotlight. Individual interviews were more reserved, sometimes short due to shyness of the student. For the most part, all students interviewed expressed positive sentiments about COSMIC club, computer science, and hope and confidence in their future. Not a single student expressed dislike of COSMIC or computer science, while some did express that their future careers were not computer science oriented. All students interviewed expressed that they would be interested in participating in COSMIC club if the opportunity was available to them again.

Text from each interview was run through a program that used TextBlob's sentiment analysis feature to produce the results seen in Tables 5.5 and 5.6.

Table 5.5 Sentiment analysis results for WLMS interviews.

school	gender	Interview type	Interview time	Interview word count	Interview polarity score	Polarity score rating
WLMS	male	individual	4:18m	383	.0923	positive
WLMS	male	individual	3:31m	267	.1819	positive
WLMS	female	individual	7:41m	670	.0500	neutral
WLMS	male	individual	2:38m	263	.2625	positive
WLMS	male	individual	3:21m	303	.2271	positive

WLMS overall sentiment: .1232 (positive)

Table 5.6 Sentiment analysis results for GFMS interviews.

school	gender	Interview type	Interview time	Interview word count	Interview polarity score	Polarity score rating
GFMS	1 male 2 female	group	14:10m	1,482	.1441	positive
GFMS	2 female	group	8:45m	1,247	.2818	positive

GFMS overall sentiment: .2069 (positive)

As seen in the two aforementioned tables, all interviews but one scored a positive polarity score, expressing a mostly positive sentiment, with one scoring a neutral polarity score. No interviews scored in the negative polarity score range. Group interviews had a higher overall polarity score, with the highest score attributed to the second group interview at Granite Falls Middle

School (GFMS) consisting of two female students being interviewed. Interestingly, this interview has a proportionally very high word count to interview time ratio, which could be attributed to familiarity between interview subjects and thus a more comfortable environment for an interview.

All students interviewed expressed interest in and/or plans of going to college. All students expressed that this instance of COSMIC club was not their first exposure to computer science, many referencing either a previous instance of COSMIC club or a robotics/engineering class they took as an elective at their school. A few students expressed that an older sibling or friend either introduced or encouraged them to try COSMIC club. Several students also mentioned a teacher who taught them about computer science or informed and encouraged them to participate in COSMIC club. Students expressed prior experience learning about robotics or writing code in school, some as early as elementary school. One student expressed that at first she tried to do the activities without watching the videos but once she began watching the videos before attempting to do the activity she got the hang of it. Another expressed that she did not need the videos, but she did watch them. Interviews were slightly hectic due to having taken place on the last day of COSMIC club, where kids were more energetic and less focused and interruptions like group photos occurred. In the future it would perhaps better serve research assistants to conduct interviews toward the end of the club session but not the very last day, or to even have a meeting day completely dedicated to interviewing.

Individual interviews were less productive in elucidating more nuanced looks into students' lives and perceptions of CS/CT. Student answers in individual interviews tended to be shorter and interviews flowed less naturally. Individual interviews were conducted in the same classroom where COSMIC was occurring, so other students were still working on projects or chatting with friends, and this could have contributed to the individual interviews being more awkward.

Overall, a group setting of at least two interview subjects seemed to have better chemistry. When the kids were comfortable, they talked more and gave a deeper insight into their lives and perceptions. The kids being comfortable with each other and knowing each other or being friends also helped the interviews flow. Group interviews also saw the brief participation of a teacher familiar with the students, which can often supplement insight provided by the students themselves [3].

Chapter 6 - Conclusions

COSMIC activities spanned three years serving nearly 700 persons through 175 distinct events resulting in a total of 16,000 hours of impact. Females comprised 51.1% of the individuals served, and 10.2% are classified as part of other under-represented groups. This impact measurement is conservative as anecdotal evidence indicates that some teachers used Scratch activities in their regular classes, but this was not quantified. Furthermore, the devices purchased to facilitate the COSMIC activities were used in many other ways and continue to provide opportunities for Caldwell County students.

COSMIC was largely successful in positively impacting Caldwell County students over the course of three years via after-school clubs, summer camps, and other computer science centric activities. Students participating in COSMIC showed growth and development in their understanding of Brennan and Resnick's CT concepts as measured by the Dr. Scratch analysis of hundreds of their projects. This analysis showed not only proficiency, but growth in the CT concepts of events, parallelism, conditionals, operators, data, loops, and sequences. The Dr. Scratch "Abstraction and problem decomposition" section of analysis maps back to the CT practices of modularization and abstraction. The structure of COSMIC and CS First themselves were an implementation of the CT practice "taking an incremental and iterative approach to problem solving." Student completed activities each week that were incrementally more complex and often built upon one another. Survey responses and sentiment analysis of interviews were indicative of the shift in perspective from passive to actively viewing technology as a medium for creation. All three measures show different aspects of the development of CT in students between the beginning and

end of COSMIC club. The success of COSMIC was also endorsed by the middle school curriculum director at Caldwell County schools [8–10].

CS First curriculum along with the Scratch programming language is an effective methodology for teaching CT concepts to primary school students like those who participated in COSMIC. Scratch is an ideal tool for teaching kids programming because it embodies the idea of a "low floor, high ceiling." This means that there is not a huge learning curve for a beginner to create working Scratch programs (low floor), but there is also the potential for a more experienced programmer to create complex projects (high ceiling) [6]. Dr. Scratch can be used to measure the demonstration of CT conceptual knowledge in Scratch projects so that growth can be monitored even without extensive computer science knowledge and experience. This opens the door for more teachers to host efforts to expose primary school students to CT.

It is important to try to understand the relationship that students have with computer science and CT, in hopes that the underrepresentation of women and minorities in STEM fields can be mitigated. Efforts like COSMIC expose kids to career options in STEM that they may not have known about otherwise and sets them up to potentially take other CS courses during their secondary and post-secondary education. Students interviewed during COSMIC showed confidence in their ability to learn CT and enthusiasm for future endeavors in which they might learn more about programming and other computer science concepts. Not only is exposure to CT beneficial in that it might help encourage more underrepresented demographics to enter STEM fields, CT is beneficial regardless of discipline and students learn concepts and practices that will benefit them in cross-disciplinary problem solving situations. This learning also better prepares them for a world in which technology is ubiquitous and their relationship and comfort with technology is essential for success.

A student's learning identity is complex and important in relation to how they view people who are computer scientists, themselves, and their learning experience in computer science [13]. Connecting is an important aspect of CT perspective, and activities like COSMIC are good venues for students to connect and experience learning about CT and computer science as a social experience. Students who perceive themselves as "computer-type" people were shown to be more likely to engage in teaching others about technology [13]. It was common during COSMIC meetings for students to help one another when needed without prompting from teachers or research assistants.

COSMIC efforts are no longer active, but its effects remain in Caldwell county. The professional development received by Caldwell Country middle school teachers is expected to precipitate further integration of CT concepts into regular curriculum in middle schools. Many of the students who participated in COSMIC have moved on to high schools in the area, with several noting their acceptance to "accelerated" programs like Caldwell Early College High School [28]. Most students interviewed or surveyed during COSMIC indicated they would be interested in participating again if they had the option to. It would be beneficial to COSMIC's efforts if data was collected regarding participation in computer science courses and/or activities in Caldwell county high schools. The expectation is that participation would be higher amongst students who participated in COSMIC activities or other computer science activities prior to high school than those who had not. Programs like COSMIC who have both female and male research assistants participating in the field also help provide role models for students in computer science. Having a role model is important for recruitment into STEM fields and even more so for retention of women [5]. The continued development of tools to automate "grading" of scratch projects, like Dr. Scratch, will make it more possible for teachers with less CS knowledge to teach CS or incorporate it into their curriculum.

Bibliography

[1] George G. Bear, Herbert C. Richards, and Paul Lancaster. 1987. Attitudes toward computers: Validation of a computer attitudes scale. *J. Educ. Comput. Res.* 3, 2 (1987), 207–218.

[2] Bryce Boe, Charlotte Hill, Michelle Len, Greg Dreschler, Phillip Conrad, and Diana Franklin. 2013. Hairball: Lint-inspired static analysis of scratch projects. In *Proceeding of the 44th ACM technical symposium on Computer science education*, 215–220.

[3] Karen Brennan and Mitchel Resnick. 2012. New frameworks for studying and assessing the development of computational thinking. In *Proceedings of the 2012 annual meeting of the American Educational Research Association, Vancouver, Canada*, 25.

[4] Quincy Brown, William Mongan, Dara Kusic, Elaine Garbarine, Eli Fromm, and Adam Fontecchio. 2013. Computer aided instruction as a vehicle for problem solving: Scratch programming environment in the middle years classroom. *Retrieved Sept.* 22, 6.1 (2013), 1.

[5] Benjamin James Drury, John Oliver Siy, and Sapna Cheryan. 2011. When Do Female Role Models Benefit Women? The Importance of Differentiating Recruitment From Retention in STEM. DOI:https://doi.org/10.1080/1047840X.2011.620935

[6] Shuchi Grover and Roy Pea. 2013. Computational thinking in K–12: A review of the state of the field. *Educ. Res.* 42, 1 (2013), 38–43.

[7] Lucas von Hollen, Jeffery Edelstein, Marcia A. Mardis, and Faye R. Jones. 2017. Middle School Computer Science Engagement Through Google CS First. In *Global Conference on Education and Research (GLOCER 2017)*, 171.

[8] James B. Fenwick Jr. and Keith Hindmann. 2015. Personal Communication.

[9] James B. Fenwick Jr. and Keith Hindmann. 2016. Personal Communication.

[10] James B. Fenwick Jr. and Keith Hindmann. 2017. Personal Communication.

[11] John H. Maloney, Kylie Peppler, Yasmin Kafai, Mitchel Resnick, and Natalie Rusk. 2008. *Programming by choice: urban youth learning programming with scratch*. ACM.

[12] John Maloney, Mitchel Resnick, Natalie Rusk, Brian Silverman, and Evelyn Eastmond. 2010. The scratch programming language and environment. *ACM Trans. Comput. Educ. TOCE* 10, 4 (2010), 16.

[13] E. M. Mercier, B. Barron, and K. M. O'Connor. 2006. Images of self and others as computer users: the role of gender and experience: Images of computer users. *J. Comput. Assist. Learn.* 22, 5 (September 2006), 335–348. DOI:https://doi.org/10.1111/j.1365-2729.2006.00182.x

[14] Jesús Moreno and Gregorio Robles. 2014. Automatic detection of bad programming habits in scratch: A preliminary study. In *2014 IEEE Frontiers in Education Conference (FIE) Proceedings*, 1–4.

[15] Jesús Moreno-León and Gregorio Robles. 2015. Dr. Scratch: a Web Tool to Automatically Evaluate Scratch Projects. In *WiPSCE*, 132–133.

[16] D. Ozoran, N. Cagiltay, and D. Topalli. 2012. Using scratch in introduction to programming course for engineering students. In *2nd International Engineering Education Conference (IEEC2012)*, 125–132.

[17] Mitchel Resnick, John Maloney, Andrés Monroy-Hernández, Natalie Rusk, Evelyn Eastmond, Karen Brennan, Amon Millner, Eric Rosenbaum, Jay S. Silver, and Brian Silverman. 2009. Scratch: Programming for all. *Commun Acm* 52, 11 (2009), 60–67.

[18] José-Manuel Sáez-López, Marcos Román-González, and Esteban Vázquez-Cano. 2016. Visual

programming languages integrated across the curriculum in elementary school: A two year case study using “Scratch” in five schools. *Comput. Educ.* 97, (2016), 129–141.
[19] Tracie M. Salinas and James B Fenwick Jr. 2015. COSMIC Year 1 Interim Progress Report.
[20] Tracie M. Salinas and James B. Fenwick Jr. 2017. COSMIC Year 2 and 3 Interim Report.
[21] David Weintrop, Elham Beheshti, Michael Horn, Kai Orton, Kemi Jona, Laura Trouille, and Uri Wilensky. 2016. Defining computational thinking for mathematics and science classrooms. *J. Sci. Educ. Technol.* 25, 1 (2016), 127–147.
[22] Jeannette M. Wing. 2006. Computational thinking. *Commun. ACM* 49, 3 (2006), 33–35.
[23] Jeannette M. Wing. 2008. Computational thinking and thinking about computing. *Philos. Trans. R. Soc. Math. Phys. Eng. Sci.* 366, 1881 (2008), 3717–3725.
[24] Scratch Wiki - Scratch Wiki. Retrieved July 8, 2019 from https://en.scratch-wiki.info/wiki/Scratch_Wiki
[25] Resources - CS First. Retrieved July 16, 2019 from https://csfirst.withgoogle.com/s/en/teachers
[26] Storytelling Lesson Plan. *Google Docs*. Retrieved May 14, 2019 from https://docs.google.com/document/d/1j7CjRC8hDL6lMzVUN1PKbmeeSaTwvna0_vYfGCa71yE/edit?usp=sharing&usp=embed_facebook
[27] Dr. Scratch. Retrieved May 21, 2019 from http://drscratch.org/
[28] CCC&TI Caldwell Early College High School. Retrieved July 15, 2019 from https://www.cccti.edu/Students/CECHS.asp

Vita

Kara Elise Beason was born in Tallahassee, Florida to Doug and Patti Beason. She graduated from Florida State University in December of 2015 with a Bachelors of Science in Computer Science and a minor in Mathematics. She began her Masters degree in Computer Science at Appalachian State University in January of 2017. In October of 2019 she moved to Greenville, South Carolina with her dog Charlie, where she continues her work as a software engineer.