

**EXPLORING THE IMPACT OF A SCHOOL-UNIVERSITY PARTNERSHIP
MODEL ON SUPPORTING COMPUTER SCIENCE LEARNING AMONG
MIDDLE SCHOOL STUDENTS**

by

Yi-Cheng Pan

An executive position paper submitted to the Faculty of the University of Delaware in partial fulfillment of the requirements for the degree of Doctor of Education in Educational Leadership

Summer 2018

© 2018 Yi-Cheng Pan
All Rights Reserved

**EXPLORING THE IMPACT OF A SCHOOL-UNIVERSITY PARTNERSHIP
MODEL ON SUPPORTING COMPUTER SCIENCE LEARNING AMONG
MIDDLE SCHOOL STUDENTS**

by

Yi-Cheng Pan

Approved:

Chrystalla Mouza, Ed.D.
Interim Director of the School of Education

Approved:

Carol Vukelich, Ph.D.
Dean of the College of Education and Human Development

Approved:

Douglas J. Doren, Ph.D.
Interim Vice Provost for the Office of Graduate and Professional
Education

I certify that I have read this executive position paper and that in my opinion it meets the academic and professional standard required by the University as an executive position paper for the degree of Doctor of Education.

Signed:

Chrystalla Mouza, Ed.D.
Professor in charge of executive position paper

I certify that I have read this executive position paper and that in my opinion it meets the academic and professional standard required by the University as an executive position paper for the degree of Doctor of Education.

Signed:

Fred T. Hofstetter, Ph. D.
Member of executive position paper committee

I certify that I have read this executive position paper and that in my opinion it meets the academic and professional standard required by the University as an executive position paper for the degree of Doctor of Education.

Signed:

Nancy Lavigne, Ph. D.
Member of executive position paper committee

I certify that I have read this executive position paper and that in my opinion it meets the academic and professional standard required by the University as an executive position paper for the degree of Doctor of Education.

Signed:

Lori L. Pollock, Ph. D.
Member of executive position paper committee

ACKNOWLEDGMENTS

I would like to thank and acknowledge my advisor, Dr. Chrystalla Mouza for giving me the opportunity to work with her and being a continuous source of inspiration, support, and encouragement throughout this endeavor. During this journey, Dr. Mouza provided me guidance, knowledge and a belief in myself as I began to grow professionally. I would also like to give thanks to my supervisor, Dr. Lori Pollock, for her valuable and constructive suggestions about the study and giving me the opportunity to work with her on the Partner4CS project. Many thanks must also go to my committee members, Dr. Fred Hofstetter, with whom I began my journey at the University of Delaware, and Dr. Nancy Lavigne for their advice and suggestions.

I am very grateful for the support and assistance of the teacher who participated in this work and her willingness to turn her classroom into a laboratory during the research process. I would also like to thank the school students and computer science undergraduates who participated in this work, as well as Partner4CS members Dr. James Atlas and Dr. Terry Harvey.

Finally, I will be forever grateful to my loving wife, Marybeth; my parents, John and Julia; my uncle and aunt, Michael and May Kuo; and my sister, Annie, for their unconditional love, support, and understanding during this process and throughout my life. Last but not least, I have also been fortunate to have friends, co-workers, and research colleagues who have enriched my graduate experience in different ways.

This work was supported by the National Science Foundation under award number: 1240905.

DEDICATION

This work is dedicated to my Lord Jesus Christ. All praise, honor and glory to my Lord Jesus Christ for His richest grace and mercy for the accomplishment of this work.

TABLE OF CONTENTS

LIST OF TABLES	ix
LIST OF FIGURES	xi
GLOSSARY	xii
ABSTRACT	xv

Chapter

1	INTRODUCTION	1
	Introduction and Background	1
	Relevant Literature.....	3
	Reforming CS Education	3
	Community-Based Approach to CS Education	4
	Defining Computational Thinking	5
	The Relationship between Computational Thinking and Programming..	6
	Measuring Computational Thinking	10
	Computer Science Content and Pedagogical Strategies	11
	Context of This Work	15
	University of Delaware’s Partner4CS.....	15
	The Setting of the Course and the Partnership Model	16
	University Course	16
	Partnerships and Field Activities	19
	Purpose of the Study and Research Questions.....	20
	Research Questions.....	20
2	RESEARCH DESIGN AND METHODOLOGY	22
	The Description of the Classroom Context.....	22
	Study Participants	23
	Data Collection	24
	Observation Data	25
	Data from CS Undergraduates	26
	Data from Partner Teacher.....	27
	Data from Middle School Students.....	27

Data Analysis	29
Data from CS Undergraduates	29
Data from Middle School Students	32
3 FINDINGS	34
Research Question 1	34
Research Question 2	36
Quality of Instructional Materials	36
Research Question 3	46
Computational Thinking Acquisition in Students’ Content Assessment	46
Computational Thinking Concept Development in Students’ Design Tasks	48
Students’ Views of Scratch	48
4 DISCUSSION AND CONCLUSIONS	53
Discussion	53
Recommendations	54
Implications and Future Research	56
Limitations	57
Conclusion	58
REFERENCES	59
Appendix	
A PRE- AND POST- CLUB QUESTIONNAIRE	64
B IRB APPROVAL NOTIFICATION DOCUMENT	68

LIST OF TABLES

Table 1	Brennan and Resnick’s CT Framework.....	9
Table 2	Framework for Equitable CS Instruction.....	12
Table 3	Seven Big Ideas of Computer Science Principles Curriculum	13
Table 4	Brief Description of the University Course	17
Table 5	Description of Undergraduate Participants in the Partnership.....	23
Table 6	Research Questions and Data Collection Matrix	25
Table 7	Prompts for Case Reports	26
Table 8	Design and Reflective Journal	29
Table 9	CS Case Report Criteria and Rubric Established from Literature Review	30
Table 10	Coding Schemes.....	31
Table 11	Roles of CS Undergraduates in Partnerships	34
Table 12	Average Mean Scores on the CS Case Report Rubric	37
Table 13	Description of Cases Enacted by Undergraduates: A Focus on Big Ideas	38
Table 14	Description of Cases Enacted by Undergraduates: A Focus on CT- Infused Pedagogy.....	39
Table 15	Description of Cases Enacted by Undergraduates: A Focus on CT Tool Selection	40
Table 16	Formative Assessment: Weekly Design and Reflective Journal	43
Table 17	Description of Cases Enacted by Undergraduates: A Focus on Assessments	45
Table 18	Dependent (Paired) T-Test of Pre- and Post- Content Assessment	47
Table 19	Pre- and Post- Scores on Scratch Knowledge Assessment Items	47

Table 20	Coding Schemes for Students' Views of Scratch	49
----------	---	----

LIST OF FIGURES

Figure 1	Scratch, Block-Based Programming Language	8
Figure 2	Scratch Clubhouse	22
Figure 3	The Slides of CS Unplugged Activity: Loops	41
Figure 4	The Exit Tickets Developed by Undergraduates	42
Figure 5	Students' Design Journal	42
Figure 6	Students' Story Board	43
Figure 7	A Group of TCS Students Invited to Participate in the CIS Showcase Event at UD.....	46
Figure 8	Scratch Project Level Analysis of Students' Pre-and Post- Design Tasks	48

GLOSSARY

Term	Explanation
<i>Alice</i>	An online block-based language environment for users to create animations and interactive media. https://www.alice.org/
<i>App Inventor</i>	An open-source web application developed by Google and maintained by MIT, allowing users to create software application for Android with a visual-object, drag-and-drop, and block-based language. http://www.appinventor.org/
<i>Arduino</i>	Arduino is a platform designed for electronics projects, consisting of a programmable circuit board and its integrated development environment (IDE). https://www.arduino.cc/
<i>BeeBot</i>	An educational robot designed for young kids. https://www.bee-bot.us/
<i>Blockly</i>	A visual block-based language developed by Google. Blockly has been adapted and integrated in many projects. https://developers.google.com/blockly/
<i>Code Monkey</i>	An online website promoting code literacy with gamification elements. https://www.playcodemonkey.com/
<i>Code.org</i>	A nonprofit organization dedicated to broadening participation and access in computer science education with online activities and comprehensive K-12 computer science curriculum. https://code.org/
<i>CS First</i>	Google's CS First, a curriculum designed to introduce kids to computer science with activities and projects. https://csfirst.withgoogle.com/en/home
<i>Dash and Dot</i>	Educational robot kits developed by Wonder Workshop, programmed by Blockly. https://www.makewonder.com/dash
<i>Edison Robot</i>	Educational robot kit programmed by barcodes or block-based language. https://meetedison.com/
<i>E-textiles</i>	Electronic textiles are wearable computing projects that enable digital components and electronics to be embedded in fabrics.
<i>Finch Robot</i>	An educational robot that works with Scratch or Snap, designed to develop students' computer science learning and computational thinking. https://www.finchrobot.com/

Hour of Code	A movement and campaign for computer science education organized by Code.org and an online educational platform with tutorials and activities for kids in coding. https://hourofcode.com/us
HTML	Hypertext Markup Language is the standard markup language for creating web pages and web applications.
Hummingbird	An educational robot that works with Scratch or Snap, designed to introduce kids to robotics and engineering. https://www.hummingbirdkit.com/
Java	A computer-programming language.
Khan Academy	Educational organization providing online interactive lessons and content for students and educators. https://www.khanacademy.org/
Makey-Makey	A circuit board allowing users to connect conductive objects with alligator clips and a USB cable to computer programs and send keyboard or mouse click inputs. https://makeymakey.com/
Mbot	Educational robot kit developed by Make block, programmed by block-based language. https://www.makeblock.com/steam-kits/mbot
Micro:bit	A tiny programmable computer built on Blockly and designed to promote learning computing and coding. http://microbit.org/
Ozobot	A robot designed for education, programmed by drawing color codes or Ozoblockly. https://ozobot.com/
Pencil Code	A block-based programming tool. https://pencilcode.net/
Python Turtle	A python feature, commanding a turtle to move around like a drawing board for beginners.
Raspberry Pi	A small single-board computer developed in the U.K. by the Raspberry Pi Foundation to promote basic computer science education. https://www.raspberrypi.org/
Scratch	A block-based programming tool developed by MIT. https://scratch.mit.edu/
Snap!	A block-based programming tool developed by Berkeley. https://snap.berkeley.edu/
Sphero	A robot designed for education, programmed by blocks or JavaScript. https://www.sphero.com/
Tynker	A block-based programming tool. https://www.tynker.com/

3D Printing

A process of creating a three-dimensional object with 3D printers and software such as Tinkercad.
<https://www.tinkercad.com/>

ABSTRACT

Recent policies and initiatives emphasized the importance of helping all students acquire Computer Science (CS) knowledge and develop Computational Thinking (CT). This study investigated the impact of a school-university partnership model on school students' CT development in the context of an after-school program. This study also examined the ways in which college CS undergraduates supported practicing teachers in the field, both through teaching practices and material development. The overall purpose of this work is to link CS undergraduates' practices from the designed partnership model to in-class support for teachers and subsequent student outcomes.

Participants included 65 school students in grades four to six who voluntarily participated in the nine-week after-school program, as well as six CS undergraduates and one practicing teacher who designed and implemented the program. Data were collected from multiple sources, including after-school program observations, undergraduate reflections, CS case reports, CS content assessments, programming products, design journals, and a teacher interview.

Results indicated that CS undergraduates actively sequenced, facilitated, and co-led a set of progressive, cohesive, meaningful, and relevant CS lessons with sound pedagogical strategies and age-appropriate CT tools and activities. School students' learning progress was also guided and built upon several formative and project-based assessments. Further, results indicated that the school-university partnership program positively influenced student learning of CS concepts, practices, and perspectives. Findings from this work can help provide guidelines for the design and implementation of effective partnership programs that help broaden participation in

computing, with attention to the ways in which field experience and CS undergraduates can help support CS education in local communities.

Chapter 1

INTRODUCTION

Introduction and Background

Over the past 10 years, growth in STEM (Science, Technology, Engineering, and Mathematics) jobs was three times greater than non-STEM jobs (Computer Science Teacher Association and Association for Computing Machinery, 2013) and current projections indicate that by 2020 there will be 9.2 million STEM jobs (Bureau of Labor Statistics, 2010). STEM occupations will provide 2.4 million job openings through 2018, including 1.1 million new job openings and 1.3 million replacement jobs due to retirement (CSTA Association for Computing Machinery, 2013). However, the U.S is facing shortages of people who either meet or are willing to receive STEM competencies. At the same time, research has shown that quantifiable STEM shortage is an important priority that needs to be addressed through a national strategy in order to sustain economic innovation and competency (CSTA Teacher Certification Task Force, 2008; Goode, 2011).

Throughout all STEM fields, computer science (CS) is the primary leading driver for job growth. More than 50% of the jobs in STEM fields projected by 2018 are in computing occupations, which means there are 4.6 million jobs waiting for those leaving college (CSTA Association for Computing Machinery, 2013). CS is one of the most in-demand degrees (CSTA Association for Computing Machinery, 2013). Technology has critical relationships between the high-tech industries, national security, long-term economy, and the ability to maintain global leadership in

innovation. There is also a critical link between CS education and economic growth (CSTA Teacher Certification Task Force, 2008; Goode, 2011). Specifically, CS provides the knowledge and skills all students need (even those pursuing non-STEM-related fields) to participate in the new global information society (CSTA Teacher Certification Task Force, 2008; Wing, 2006).

The U.S. education system, however, is not educating enough students with STEM-capable competencies, especially in CS, to keep up with demand both in traditional STEM occupations and other new occupations that need similar competencies (National Research Center, 2010; Sengupta, Kinnebrew, Basu, Biswas, & Clark, 2013). Specifically, only nine U.S. states are counting CS courses as a core requirement in K-12 education. According to the office of the Press Secretary (2016) “by some estimates, just one quarter of all the K-12 schools in the United States offer CS with programming and coding, and only 28 states allow CS courses to count towards high-school graduation, even as other advanced economies are making CS available for all of their students” (p. 1). As a result, educators are facing a responsibility to ensure all students have opportunities to learn CS principles and computational thinking (CT) practices that are relevant to their future career and lives (Barr, Harrison, & Conery, 2011; Sengupta et al., 2013; Wing, 2006).

It is clear, that the U.S. K–12 educational system continues to marginalize CS education (CSTA Teacher Certification Task Force, 2008; Goode, 2011). Although there is clearly a demand, there is currently no way to ensure that students learn the best-prepared content and skills they need from well-trained qualified CS educators (CSTA Teacher Certification Task Force, 2008). Federal, state, and local policies governing teacher certification also result in barriers to, rather than support for,

teaching and learning in CS. According to the CSTA Teacher Certification Task Force (2008), the U.S. needs clear, consistent, and rational policies governing what teachers need to know, along with similar policies on how to systematically train teachers. In the absence of teacher preparation and support, the U.S. will be under-producing STEM workers for the next generation.

Relevant Literature

Reforming CS Education

Many governments around the world are working on improving their CS education and curriculum. The British government is changing its CS education and CS has become a part of England's primary school curriculum. Other countries such as Australia, Denmark, Israel, New Zealand, and Germany have also updated their CS high-school curricula (A is for Algorithm, 2014).

In the U.S., President Donald Trump's new CS campaign builds in some ways on previous work by his predecessor, former President Barack Obama, who sought to boost coding education through initiatives like *Computer Science for All*. Many tech industry leaders such as Google, Apple, Amazon and Microsoft have also provided funding to begin developing resources and models. *CS for All* called for a 4.1 billion budget targeting the following goal: "children from kindergartens through high schools need to learn Computer Science and be equipped with the CT skills they need to be creators in the digital economy, not just consumers, and to be active citizens in our technology-driven world" (Whitehouse.gov, 2016, p. 1). Further, this initiative aims to provide professional development for teachers with high-quality instructional materials.

The National Science Foundation and its *CS 10K* project aims to address teacher preparation explicitly and prepare 10,000 teachers in 10,000 high schools. The target date for having these teachers in place, which has been extended, was 2015. The project is a large-scale, collaborative effort bringing communities together with “the goal of systematically changing the scale, curriculum, and pedagogy of teaching computer science at all levels,” with a particular focus on CS in U.S high schools as well as introductory computing at the college level (for more information, <http://cs10kcommunity.org/>). The NSF also launched a new program called *Computing Education for the 21st Century* (CE21) to help leverage *CS10K* effort.

The Computer Science Teacher Association (CSTA) released the revised K-12 Computer Science standards (Computational Thinking—Teacher Resources second edition, 2017). The standards provide a framework and foundation for K-12 CS education. The Association for Computing Machinery (ACM) and CSTA also launched a new K-12 advocacy coalition called *Computing in the Core* to establish partnerships. Further, many high schools implemented an introductory Computer Science course called *Exploring Computer Science* (<http://www.exploringcs.org>) and adapted a new approach for AP CS through *Computer Science Principles* (National Science Foundation, 2014).

Community-Based Approach to CS Education

CS Education reform to date has involved community building, engagement, commitment and resources. Yet, Repenning, Webb, and Ioannidou (2010) indicated that CS education has not received much attention at the middle school level. For instance, programming is not commonly seen in middle school curricula or in schools. Therefore, informal learning with coding skills has become available through many

online communities, makerspaces, and local after-school computer clubs (Matias, Dasgupta, & Hill, 2016). Research suggests that investing in partnerships and implementing extracurricular programs, such as summer camps or after-school programs are strongly recommended to introduce CT and CS concepts to school students and broaden participation in the field (Maloney, Kafai, Resnick, & Rusk, 2008; PCAST, 2010; Repenning et al, 2010).

Defining Computational Thinking

With many initiatives and recent policy emphasizing the importance of helping all students be equipped with CS knowledge and skills, CT has emerged as an essential literacy that has been frequently included as part of 21st century skills across disciplines (NRC, 2010). In particular, CT is considered a way to promote CS education across the curriculum.

CT was pioneered by Jeannette Wing (2006). Wing (2008) introduced CT as “solving problems, designing systems, and understanding human behavior, by drawing on the concepts fundamental to computer science” (p. 3717). She suggested that CT is a fundamental skill of analytical thinking for everyone, not just for computer scientists. CT is a problem-solving method and defined as “the thought processes involved in formulating a problem and expressing its solution in a way that a computer—human or machine—can effectively carry out” (Wing, 2006, p. 33).

CT is also included as a key element in the newly released National Educational Technology Standards for Students from the International Society for Technology in Education (ISTE, 2016). ISTE defined the purpose of CT as follows: “Students develop and employ strategies for understanding and solving problems in

ways that leverage the power of technological methods to develop and test solutions” (<http://www.iste.org/standards/standards/for-students-2016>).

The International Society for Technology in Education (ISTE) in collaboration with the Computer Science Teachers Association (CSTA) published a list to define CT characteristics (Barr & Stephenson, 2011; CSTA & ISTE, 2011). These include, but are not limited to:

- Formulating problems for use with a computer to facilitate the solution;
- Logically organizing and analyzing data;
- Representing data through abstractions;
- Automating solutions through algorithmic processes;
- Identifying, analyzing and implementing possible solutions, as the most efficient and effective combination of steps and resources; and
- Generalizing and transferring this process to a variety of problem areas.

The Relationship between Computational Thinking and Programming

Programming has received great attention worldwide thanks to block-based environments, such as Scratch. Programming not only allows students to create programs with computers, but also easily connects with robotics and computational tools (i.e., *Sphero*, *Micro:bit*, *Ozobot*, *Makey-Makey*, *Finch Robot*, *Mbot*, *BeeBot*, *Edison bot*, *Dash and Dot*, *Raspberry Pi*, *3D Printing*, *Hummingbird*, *Arduino*, *Python turtle*, *E-textiles*). Research has indicated that programming games provide a great motivation for students and serves as an introduction to CS, offering a context that involves many CT elements (Repenning et al, 2010; Shute, Sun, & Asbell-Clarke,

2017). As a result, many early introductory CS or CT-infused curricula were designed around visual block-based programming language environments such as *Scratch*, *Alice*, *Blockly*, *MIT App Inventor*, *Tynker*, and *Snap!* These tools focus on providing young learners with fun, engaging learning experiences with CT. Some online introductory programming curriculum resources—*Code.org*, *Khan Academy*, *Google’s CS First*, *Pencil Code*, or *Code Monkey*—have already become part of many children’s initial programming experiences either in schools, in after-school clubs or at home. *Code.org* is a nonprofit dedicated to broadening participation and access in CS education with k-12 CS curriculum. *Hour of Code*, a movement and campaign organized by *Code.org* offers an online educational platform for many students to try coding for one hour with age-appropriate projects, selectable activities, and interactive tutorials (for more information, <https://hourofcode.com/us>).

Scratch. Over the past several years, *Scratch*, an online educational programming tool launched in 2007 by MIT Media Lab, has generated great attention by providing an easy-to-use and user-friendly online programming environment. Especially in CS education, Scratch has been widely adopted in K-12 settings. With Scratch, users are able to create their own digital stories, animations, games, music, and interactive media. Scratch also enables users to share, comment on, and remix projects online, offering an environment for design-based learning (Brennan & Resnick, 2012). This program is frequently called low-floor high ceiling (Brady et al., 2017), to indicate that is easy enough for novice users to create simple projects while simultaneously allowing advanced users to create more complex projects. Block-based and color-coded elements allow users to see errors right away without worrying about syntax (see Figure 1). Further, an online platform developed in conjunction with this

tool, called *ScratchEd*, provides a community resource for teachers who are interested in sharing materials and experiences related to teaching and learning with Scratch (<https://scratch.mit.edu>).

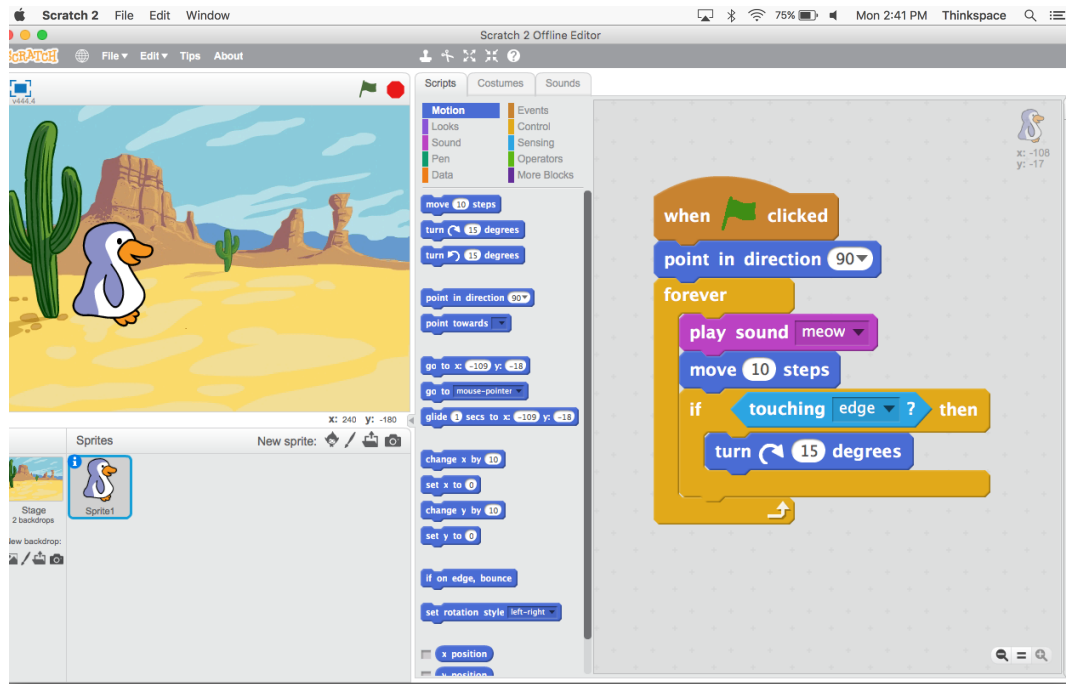


Figure 1 Scratch, Block-Based Programming Language

Brennan and Resnick (2012) identified three specific dimensions of CT observed through project creation in Scratch, which include: (a) *Computational Concepts*: sequences, loops, parallelism, events, conditional, operators, and data; (b) *Computational Practices*: being incremental and iterative, testing and debugging, reusing and remixing, and abstracting and modularizing; and (c) *Computational Perspectives*: expressing, connecting, and questioning (see Table 1).

Table 1 Brennan and Resnick's CT Framework

Computational Concepts	
Sequences	A specific task or action is expressed as a series of steps or instructions that can be executed by the computer.
Loops	A mechanism used to run the same sequence of instructions multiple times.
Events	An important component of interactive media where one thing triggers another thing to produce actions.
Parallelism	Sequences of instructions happening simultaneously within a single object or across objects.
Conditionals	Multiple outcomes are determined based on certain conditions.
Operators	Operators performs numeric and string manipulations including mathematical, logical, and string expressions, such as addition, subtraction, multiplication, division, functions, sine, exponents, concatenation and length of strings.
Data	Data involves variables and lists to store, retrieve, and update values, numbers and strings.
Computational Practices	
Being incremental and iterative	Iterative cycles of imagining and developing. An adaptive process of trying and changing based on new ideas, solutions and experiences.
Testing and debugging	Building projects through trial and error and developing strategies.
Reusing and remixing	A collaborative practice of getting inspired by, accessing to others' work and building upon the projects.
Abstracting and modularizing	A practice of design and problem solving: putting together collections of smaller solvable parts to create something complex.
Computational Perspectives	
Expressing	Expressing ideas and self by designing and creating computing products.
Connecting	Establishing partnerships, collaborations, and developing learning and creativity through access to networks and interactions with others in social practices.
Questioning	Learners are empowered to ask questions and make sense of computational artifacts by utilizing technologies.

Measuring Computational Thinking

In recent years, researchers have been giving CT increased attention. Yet, measuring CT development is still a remaining challenge because assessments of CT remain under-developed and under-researched (Yadav et al., 2015). Recently, however, researchers have started developing a few assessments of CT, particularly centered on programming. Specifically, since Scratch is a widely-adopted visual programming language, some authors focused on establishing assessments centered on Scratch's built-in data. Other authors, proposed frameworks to assess the development of CT in the context of designing projects on Scratch as well as specific assessment strategies such as (a) *Project Analysis*, (b) *Artifact-Based Interviews*, and (c) *Design Scenarios* (Brennan & Resnick, 2012).

Moreno-León and Robles developed an online program called “Dr. Scratch” which can analyze and score Scratch projects automatically. *Dr. Scratch* examines constructs of presence and absence of computational concepts and scores Scratch projects along seven dimensions that include abstraction and problem decomposition, parallelism, logical thinking, synchronization, flow control, user interactivity, and data representation (Moreno-León & Robles, 2015).

The Association for Computing Machinery (ACM) and the Computer Science Teacher Association (CSTA) emphasized that a factor limiting CT in schools is the lack of assessments for teachers. Much research to date has concentrated on traditional classroom assessment strategies (e.g., multiple-choice assessment). One factor that makes CT difficult to measure is that CT focuses on the thought process, not on the end product or artifact alone. Ahonen and Kankaanranta (2015) recognized that “think-aloud” could be an effective way to assess students' CT process, such as breaking down problems and developing algorithms. Think-aloud data can be

collected through (a) a *concurrent* method where students unveil their cognitive process by thinking aloud while completing tasks, and (b) a *retrospective* method in which students describe their metacognitive process of solving a problem after they complete the task (Ericsson & Simon, 1993). Between these two methods, *concurrent* can more accurately assess students' ability. However, sometimes young students have difficulty articulating their thought processes. Therefore, Mueller, Becket, Hennessey, and Shodiev (2017) also proposed a set of teacher verbal protocols to help in the assessment of students' CT process. This set includes guiding questions for each CT category based on the constructs informed by Brennan and Resnick (2012) and Csizmadia et al. (2015). By asking questions as a communication tool, teachers will be able to understand students' CT ability during conversations and pinpoint strengths and weaknesses.

Computer Science Content and Pedagogical Strategies

Developing programming environments is necessary but not sufficient for broadening participation in computing. Curriculum revisions, rigorous CS curriculum standards, and teacher preparation with renewed vision of CS pedagogy remain a more critical part of the reform. Many newly designed curricula and new pedagogical approaches are underway (Repenning et al., 2015; Shah et al., 2013; Webb, Repenning, & Koh, 2012), yet according to Shah et al. (2013) pedagogy is critical in broadening student participation in CS. Specifically, Shah et al. proposed a framework for equitable CS instruction, which consists of four dimensions: (a) Access to rich course content, (b) Quality instruction, (c) Productive peer relationships, and (d) Identities as computer scientists (see Table 2 for a description of each).

Table 2 Framework for Equitable CS Instruction

Four Dimensions	Pedagogical Practices
Access to Rich Course Content	Emphasizing Multiple Solutions Using Metaphors to Introduce Concepts Debugging by “Acting Out”
Quality Instruction	Tracking Student Progress Customizing Teaching Plans for Individual Students Using iClickers for Formative Assessment
Productive Peer Relationships	Exposing Students to a Diverse Set of Computer Scientists Managing Public Displays of Status Encouraging Connections to “Out-of- School” Identities
Identities as Computer Scientists	Strategically Partnering Students Encouraging and Structuring Peer Interaction Modeling Ideal Peer Collaboration

In terms of rich course content, in the last few years, many CS curricula were established, such as *Exploring Computer Science* (ECS), and *CS Principles* (CSP), which emerged to provide a framework of seven big ideas in computing. These include: *Creativity, Abstraction, Data and Information, Algorithms, Programming, The Internet, and Global Impact* (see Table 3 for a description of each). These principles focus on key CS constructs that can be integrated into an existing curriculum, rather than coding only. The big ideas include the relevant impact of computing on society and emphasize creativity (Cuny, 2012). Further, the CS Principles framework emphasized six CT practices, which can be applied throughout the curriculum: (1) Connecting Computing, (2) Creating Computational Artifacts, (3) Abstracting, (4) Analyzing Problems and Artifacts, (5) Communicating, and (6) Collaborating (College Board, 2014).

Table 3 Seven Big Ideas of Computer Science Principles Curriculum

Seven Big Ideas	Brief Description
Creativity	Computing is considered as a creative activity and leads problem-solving, innovations and exploration. Students utilize tools to create computational artifacts.
Abstraction	Abstraction is a strategy of problem-solving, which reduces detail to focus on relevant concepts. Students learn how to use models and simulation to simplify complex topics and identify patterns.
Data and Information	Students manage, process, interpret and visualize data by utilizing computational tools to create new information knowledge.
Algorithms	Algorithms are fundamental aspects in computing and everyday tasks. Students develop solutions to computational problems.
Programming	Students select and learn a variety of appropriate programming languages to create software and artifacts based on projects and problems.
The Internet	Communication and collaboration are supported and enabled by the networks and systems. Students explore how the Internet operates and analyze the implementation of computational solutions.
Global Impact	Students understand how computing innovations changed the way people work and live and leads to new understandings, disciplines and discoveries in the world.

A variety of pedagogical strategies have also been introduced in CS instruction. Among teaching strategies, utilizing *kinesthetic* activities or metaphors has been widely utilized. In the context of CS, these activities are called *CS unplugged* (<https://csunplugged.org/en/>) and have been used to introduce CS concepts without computers. *Process oriented guided inquiry* (POGIL), rooted in learner-centered *constructivism* (Yager, 1995), is another pedagogical strategy that allows students to work in teams on designed guided inquiry materials to actively construct and develop

content knowledge through a learning cycle: *exploration, content invention and application*. POGIL can often be employed in *problem-based approaches* and *open-ended projects* where teachers act as facilitators. Further, *Project-based learning* focuses on *constructionism pedagogy*, advocating the importance of actively learning by doing. Papert's *Constructionism* (1991) emphasizes how knowledge and ideas get formed, and transformed, shaped and expressed through different media, in specific contexts and processed in different people's minds. Therefore, the CS education reform is often connected with the *Maker Movement* (or so-called *Hackerspace, Makerspace, and Fablab*) where students explore knowledge through meaningful hands-on learning in the context of designing, making and inventing.

Other teaching strategies emphasize peer interaction and differentiated instruction. *Pair programming* is commonly adopted to support peer interaction, where two students work together with one student serving as driver and the other as navigator. Research has indicated that pair programming is an effective approach to motivating beginning programmers (Berland & Lee, 2011; Denner & Werner, 2007). Differentiated instruction occurs when students' levels vary within a module. Students with prior experience are provided with additional exercises in advanced challenges. In the context of CS education, students can be encouraged to incorporate multiple CS concepts (i.e., loops, conditionals and variables), update and remix projects, or proceed to the next levels of online tutorial-guided activities.

Both *formative and summative assessments* are also key to successful and effective teaching of CS (Grover, Pea, & Cooper, 2016; Mouza, Marzocchi, Pan, & Pollock, 2016). Sustained reflection on students' project design, answering guiding

questions or prompts in a journal can help teachers to collect learning data and adjust lesson planning. Students can also self-reflect on their learning and receive feedback.

Context of This Work

Delaware's Science, Technology, Engineering and Math (STEM) Council (2012) noted "the need for content-trained STEM teachers, particularly in engineering and technology education" (p. 2). Similar to other states, Delaware is facing the challenge of improving computing education beyond basic keyboarding literacy.

University of Delaware's Partner4CS

The University of Delaware is highly aware that both pre-and in- service teachers and students in all levels need more preparation in CS education and computational knowledge. As a result, a team consisting of professors from the School of Education and the Department of Computer and Information Science established the Partner4CS project, with funding from the National Science Foundation (under projects of CS10K, CE21, and STEM+C). Partner4CS aims to broaden participation in computing through strategic partnering. It includes four major efforts as described by Pollock, Mouza, Atlas and Harvey (2015), including:

1. Sustainable, high-quality professional development for teachers. Specifically, the research team has developed an annual summer teacher professional development institute endorsed by Delaware's Department of Education. Teacher participation in one of the professional development tracks, *Computer Science Principles*, qualifies participants to teach the CSP AP course as part of the CS pathway in high school career and technical education established in the state of Delaware.

2. A field experience university course as the primary vehicle that partners undergraduate students with practicing teachers in the field for on-going support.
3. Strong partnerships with local school districts, teachers and STEM leaders in formal and informal settings. Building on previous efforts, the team has established partnerships with local libraries, the Boy & Girls club, and school districts serving under-represented students. Further, the team also helped to establish the CSTA chapter in Delaware and organize state-wide *Scratch Days* (<https://day.scratch.mit.edu>).
4. Policy changes. In collaboration with the Delaware's Department of Education, the team helped establish a pathway for CS at the high school level and organized a Summit for CS Education in Delaware in 2017, for stakeholders and community members (teachers, leaders, school districts, and parents) who are interested in computing education. This summit built a network to support the teaching of computing at the state level. In 2017, Delaware Governor Carney also signed a bill requiring all public high schools to teach at least one CS course by the 2020-2021 academic year (for more information, <https://www.udel.edu/partner4cs>).

The Setting of the Course and the Partnership Model

University Course

The primary focus of this study is the partnership model and its university-community field experience course, titled “Field Experiences in Teaching Computing” at the University of Delaware. This field-based course serves as the primary vehicle to

partnerships, and has been offered for eleven consecutive semesters since 2013. To date, approximately 65 undergraduates have enrolled in this course.

The course involves on-campus training in preparation for successfully integrating CT practices in middle/high school classrooms or after-school settings. The objective of the field experience course is to help CS undergraduates: (a) develop new technical and teaching skills, (b) improve their communication and leadership skills, (c) participate in service to the local school communities by engaging school students with computing, and (d) reflect on their learning experiences (Pollock et al., 2015).

The topics and modules in the course are designed around four main areas, including (a) CS tools; (b) CS Pedagogy; (c) CS curriculum, trends and standards; and (d) Reflection and communication (see Table 4).

Table 4 Brief Description of the University Course

Key Areas	Specific Activities
Computer Science Tools	Lab assignments (robotics and technologies) Scratch practices
Computer Science Pedagogy	Lesson planning Effective lesson and learning environment design Teaching strategies Classroom management CS unplugged activities Learning theories
CS Curriculum, Trends and Standards	Broadening participation in computing CS impacts CSTA standards Computer science principles
Reflection and Communication	Weekly report and feedback Reflective journal Background clearances Dress code Communication before, during and after field placement

At the time of the study, approximately fifty percent of total instruction time in this semester-long course was designed to provide modules that focused on CS-related content skills and resources, especially application in K-12 CS education. These topics included Scratch introduction and practice, lab assignments on exploring computational tools (i.e., Finch Bot, Makey-Makey), CS curriculum and standards (i.e., CSP, ECS).

Forty percent of designed modules in this course intended to expose students to pedagogical strategies, classroom management, lesson plan creation, and effective lesson design and delivery. Specifically, participants were given opportunities to practice and lead a mock CS unplugged activity in class, which allowed them to discuss their strengths and weaknesses.

Another key component embedded in this course is weekly journals which provide CS undergraduates opportunities to reflect on and document their experiences, teaching approaches and strategies. Participants' interpretation and reflection on classroom experiences is intended to promote learning from practice (Kolodner, 2006). On a weekly basis, CS undergraduates reported to their peers and lead faculty what happened in their teaching, shared their experiences, and received feedback from peers and instructors. A number of learning opportunities actually occurred in the process of receiving feedback during the debriefing meetings in the field and on-campus meetings with faculty during weekly updates.

A successful application of the model relies on the solid foundation of effective and persistent communication among CS undergraduates and teachers. The partnership in fact, built on logistics happening in the very early portion of the course, which include: (a) emphasis on professional communication (i.e., emails, holiday

break, schedules) and appearance (i.e., dress code) before, during, and after the field placement; and (b) setting up regular teacher-undergraduate meetings. CS undergraduates first contacted their partner teacher to introduce themselves and scheduled their first meeting, and then requested sit-in classroom observation to gain a better understanding of their assigned classroom, including classroom management, technologies available, and students' prior experience. This background information is important before they formally start to work with their assigned teacher on lesson preparation and delivery.

Partnerships and Field Activities

Teachers who participated in the summer professional development program offered by Partner4CS are eligible to receive on-going support. The Partner4CS team first sent out invitations to participating teachers and asked them to submit requests of interest. A member of the team then partnered undergraduates in the field experience course with practicing teachers. To date, the team has established partnerships in nine school districts with over 20 sites. Approximately 500 students in Delaware in a variety of partnerships such as libraries, after-school computer club programs, or regular technology classes benefited from CS instruction offered through Partner4CS undergraduates.

CS undergraduates could take the course for one to three credits. Participants who took three credits were expected to spend at least three hours per week in the field. Generally, undergraduates were assigned to an appropriate placement in groups of two to three peers with someone providing transportation. Field placements could vary based on grade levels that spanned from primary schools to high schools, student population and demographics (i.e., ethnicity, SES, single-gender schools), settings

(informal like libraries, clubs or regular classrooms), as well as content and technologies ranging from introductory Scratch, HTML to CS AP Java.

Purpose of the Study and Research Questions

A number of studies report a shortage of qualified STEM educators and workers (CSTA Association for Computing Machinery, 2013). Further, recent policies indicate that all students should be equipped with computational knowledge, thinking, and skills (Wing, 2006). To accomplish this goal, changes must be made across the entire computing education system.

Therefore, the purpose of this study is three-fold. First, it aims to describe the specific roles served by CS undergraduates in the context of the Partner4CS field experience model. Second, it seeks to examine the quality of the instructional materials they prepared to support partner teachers and students. Third, it aims to link CS undergraduates' practices from the designed partnership model to in-class support for teachers and subsequent student outcomes. The work focuses on a partnership with a single school and teacher to provide an in-depth investigation. Findings from this work can help provide guidelines for the design and implementation of effective partnership programs, with attention to the ways in which field experience and CS undergraduates can help support CS education in local communities.

Research Questions

1. What specific roles did CS undergraduates assume in their partner school?
2. How was the quality of the instructional materials created by CS undergraduates in the context of the school-university partnership? How were CT concepts and practices implemented?

3. What computational concepts did middle school students acquire as a result of working with CS undergraduates and their teacher through a school-university partnership?

Chapter 2

RESEARCH DESIGN AND METHODOLOGY

The Description of the Classroom Context

This work examined one partnership established through Partner4CS in an after-school program at a K-8 school, Town Charter School (TCS). At the time of the study, the partnership had been in place for five years. The school's technology teacher (Ms. Sharon), established her nine-week after-school computing program for students in grades four through six. Each semester, two to four CS undergraduates partnered with Ms. Sharon for the 70-minute-long computing program in which students met once a week. The online introductory programming tool, *Scratch*, was utilized as a main tool to introduce CT concepts and skills, and thus the program was called Scratch Club (see Figure 2).

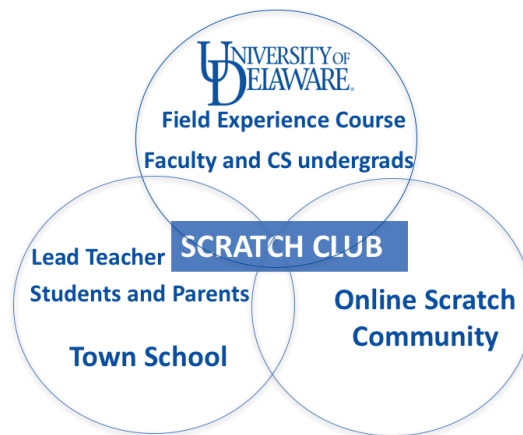


Figure 2 Scratch Clubhouse

TCS Student Population. TCS is a suburban K-8 school, enrolling approximately 1,350 students. School demographics indicated that the student body was 67.5% White, 11.6% Asian, 11.1% African American, 4.2% Hispanic, and 5.5% multi-racial. Approximately 16% of the entire student population qualified as low-income.

Study Participants

CS Undergraduates. The participants included six CS undergraduates enrolled in the course CISC 357 “Field Experience in Teaching Computing” over a period of two semesters: Spring I (From Feb. to May) and Fall II (From Sept. to Dec.). These students, three females and three males, were assigned to partner with Ms. Sharon at TCS as shown on Table 5.

Table 5 Description of Undergraduate Participants in the Partnership

Name	Semester	Gender	Background
Beth	Spring I	Female	CS major: Senior
Mark	Spring I	Male	CS major: Senior
Jason	Spring I	Male	CS major: Junior
Mary	Spring I	Female	CS major: Junior
David	Fall II	Male	CS major: Sophomore
Lauren	Fall II	Female	CS major: Sophomore

During this partnership, Ms. Sharon co-taught the nine-week after-school computer club respectively with four CS undergraduates in Spring I and with two CS undergraduates in Fall II.

School students. The sample of this study also consisted of a convenience sampling of 65 upper elementary/middle school students (N=65, 41 boys and 24 girls) in grades four through six who participated in the Scratch club in Spring I or Fall II.

Data Collection

To answer the research questions (see Table 6), this work employed a mixed method approach. Specifically, both qualitative and quantitative data were collected from multiple sources including: (a) CS *case reports* enacted by CS undergraduates; a *case report* consists of a lesson plan, its associated materials, and CS students' self-reported description of this lesson; (b) CS undergraduates' reflective journals; (c) partner teacher interview; (d) middle school students' pre- and post- content assessments; (e) pre- and post- design tasks developed by middle school students during their participation in the Scratch club; (f) middle school students' design and reflective journals; and (g) observations of the course and after-school Scratch club meetings.

Table 6 Research Questions and Data Collection Matrix

Question 1: What specific roles did CS undergraduates assume in their partner school?		
Source of Data	Data Analysis	Information Purpose
CS Undergraduates Partner Teacher	CS undergraduates weekly reflective journals Classroom and field observations Partner teacher interview	This information was used to examine the roles of CS undergraduates in the field.
Question 2: How was the quality of the instructional materials created by CS undergraduates in the context of the school-university? How were CT concepts and practices implemented?		
Source of Data	Data Analysis	Information Purpose
CS Undergraduates Lead Teacher	CS undergraduates weekly reflective journals Collection of case reports Classroom and field observations Partner teacher interview	This information was used to examine the materials in which CS created to help support teachers in the field with an additional focus on how these were implemented.
Question 3: What computational concepts did middle school students acquire as a result of working with CS undergraduates and their teacher through a school-university partnership model?		
Source of Data	Data Analysis	Information Purpose
Students in grades 4-6	Pre- and post- content assessment Pre- and post- design tasks Design and reflective journals	This information was used to examine whether the partnership model helps middle school students acquire computational concepts.

Observation Data

Observations of Course and the After-School Scratch Club Meetings. All on-campus course meetings (N=30) and after-school Scratch club sessions (N=24) were observed and documented. The author recorded key CS topics addressed,

instructional strategies and learning activities observed, assessments utilized, and technologies used by the students and the teacher in the club.

Data from CS Undergraduates

CS Case Reports Developed by CS Undergraduates. A case report consists of a brief lesson plan, associated instructional materials and artifacts developed by six CS undergraduates, and self-reported descriptions and reflections of the lesson's implementation. CS undergraduates were provided with prompts to report demographic data and how they implemented lessons each week (see Table 7 below). All case reports (N=24) throughout the academic year were collected in order to examine how CS undergraduates supported their partner teacher in the field.

Table 7 Prompts for Case Reports

Aspects	Prompts
Demographic data	<ul style="list-style-type: none"> • When and where did you teach/facilitate this lesson? Which class? What grade levels? • How many students were in the class? (record the number of male and female students).
Role in the field	<ul style="list-style-type: none"> • What type of support did you provide: (a) observed, (b) assisted the teacher, (c) led a classroom activity, (d) taught a full lesson as the primary instructor.
Lesson planning and implementation	<ul style="list-style-type: none"> • What types of activities did you implement? (CS unplugged, programming etc.). • What CS Principles did you cover? • What is your plan for the next lesson? • What materials or technologies do you need to proceed to the next step?
Reflection	<ul style="list-style-type: none"> • What went well (describe your successes)? • What would you change for the future? • What, if anything, surprised you? • What questions do you have for us?

CS Undergraduates' Reflective Journals. Participants were required to submit their weekly reflective journal entries through a blog format in the University's Learning Management System (LMS), *Sakai*. Weekly reflective journal entries (N=68) maintained by the six CS undergraduates during the partnership period were collected for analysis. The length of each entry varied. Among the six participants, the average length of each entry was approximately 400 words. This information helped identify the ways in which the undergraduates supported their partner teacher, the rationale behind the design of their instructional materials and lesson plans, and their reflections regarding the outcomes of their lessons.

Data from Partner Teacher

Partner Teacher Interview. An interview (N=1) was conducted with the partner teacher, Ms. Sharon, to discuss the ways in which the partnership model helped the implementation of CS lessons. The interview questions included the following themes: (a) The nature of implementation and design in the partnership model; (b) Expectation and feedback for CS undergraduates; (c) Instructional approaches and strategies regarding CS concepts; (d) Teaching beliefs and perspectives; and (e) The outcome of students' performance.

Data from Middle School Students

Students' Pre-and Post- Content Assessments (Spring I & Fall II). To understand changes in students' knowledge of CS concepts, a pre- and post- multiple-choice Scratch knowledge assessment was administered both at the beginning and the end of the club in these two semesters (N=65). The assessment includes 10 questions

examining students' knowledge of CS concepts associated with Scratch programming (Ericson & McKlin, 2012; see Appendix A).

Pre-and Post- Design Tasks Developed by Middle School Students in the Scratch Club (Fall II). All participating students in Fall II (N=23) were asked to develop a story, animation or a drawing in 20 minutes using Scratch at the beginning and end of their participation in the after-school program. A total of 46 projects were collected. The following prompt was used:

Show off the things you know about Scratch by creating a story, an animation, or a drawing using the cat or another sprite. Be creative. (It is ok, if you don't know a lot of things yet).

Middle School Students' Views of Scratch from Design and Reflective Journals (Fall II). Students were provided with blue notebooks at the beginning of their participation in the after-school Scratch club, that acted as journals. In those journals students drafted ideas for projects they wanted to complete in Scratch, answered given questions, and reflected on their learning after each meeting day (see Table 8). Design and reflective journals maintained by students throughout the duration of the after-school program in Fall II were collected (N=27). A specific question "*How would you introduce Scratch to your friends?*" was asked at the beginning and the end of the club. A total of 54 responses were collected under the given prompt and were used to document students' understandings of Scratch at the beginning and end of their participation in the Scratch club.

Table 8 Design and Reflective Journal

Sessions	Prompts/ Tasks
Week 2	How would you describe Scratch to a friend? (Pre)
Week 3	Storyboarding: Create a storyboard of their project.
Week 4	1. List three ways you experience loops in real life. (e.g., going to
Week 5	sleep every night). For Advanced Students:
	2. What are different ways of increasing difficulty in a game?
	3. Which extensions did you add to your game project?
Week 6	1. What is something that works well or you really like about the project?
	2. What are you most proud of? Why?
	3. What is something that is confusing? How did you get unstuck?
Week 7	1. Describe the final project you want to create.
	2. List the steps needed in order to create your project.
	3. What might you need help with in order to make progress?
	4. Editing storyboard.
Week 8	How would you describe “variables” to your friends?
Week 9	Design a Fall-theme Scratch project on your journal and share your ideas.
Week 10	How would you describe Scratch to a friend? (Post)

Data Analysis

Data from CS Undergraduates

CS Case Reports Enacted by Undergraduates. The CS case reports enacted by undergraduates were collected at the end of each semester. All associated lesson plans, materials, and artifacts were labeled and stored with short descriptions. The instructional materials were uploaded to a folder in Google Drive and scored by the investigator using a set of criteria established through the literature review (see Table 9). The criteria were established around *How People Learn* (Bransford, Brown, & Cocking, 2000) in order to capture the spirit of *Learner-centered, Knowledge-*

centered, Community-centered and Assessment-centered learning environments. The rubric also utilized and integrated existing standards for CS curricula, such as CS principles. Each of the four criteria received a numerical score from 1 to 4. A score of 1 indicates failure in satisfying the criterion, while a score of 4 indicates full success in satisfying the criterion. The author and a co-rater scored all lesson cases. The initial inter-rater reliability was calculated at 85%. All discrepancies were discussed until a 100% agreement was reached.

Table 9 CS Case Report Criteria and Rubric Established from Literature Review

Criteria	Descriptions
Standard-Based Learning Objectives and Meaningful Topics	<ul style="list-style-type: none"> • Learning objectives are aligned to standard-based CT concepts.¹ • CT concepts and vocabulary are clearly and correctly presented and involve at least one of the big ideas.² • Topic selection is meaningful and relevant to students and connected to prior knowledge or lessons.
CT Practices-Infused Environments and Instructional Strategies	<ul style="list-style-type: none"> • CT practices and perspectives³ are encouraged and presented through chosen pedagogy method and designed environments, that include access to identity, scaffoldings or differentiated instruction.

¹ Brennan and Resnick (2012); CSTA& ISTE (2012)

² CS Principles (College Board, 2014)

³ Brennan and Resnick (2012); College Board (2014)

Age-Appropriate CT Technology and Activity Selection	<ul style="list-style-type: none"> • A variety of CT-infused tools, activity or materials were identified and accessed to support learning goals and CT concepts and be compatible with instructional strategies.
Formative and Summative Assessment	<ul style="list-style-type: none"> • Selected assessment is effective and appropriate, which includes design-based or problem-based projects.

CS Undergraduates' Blog Entries from Reflective Journals. The data from CS undergraduates' reflective journals were analyzed qualitatively to identify emerging patterns (Hatch, 2002). The coding process mainly focused on the following topics, including (a) bringing CS presence, motivation, inspiration and excitement into school; (b) connecting CS with other school subjects; (c) teaching strategies and planning; (d) producing lesson plans and materials; (e) relationships with the partner teachers; (f) lab help with emotional support and technical concepts; and (g) expectation and feedback from the practicing teachers. The initial codes were then categorized into three parent codes (see Table 10).

Table 10 Coding Schemes

Parent Codes	Initial Codes
Technology Consultants	<ul style="list-style-type: none"> • Expectation and Feedback from the practicing teachers • Lab help with technical support
Facilitators and Co-Teachers	<ul style="list-style-type: none"> • Producing lesson plans and materials • Teaching strategies and planning • Connecting CS with other school subjects
Mentors and Ambassadors	<ul style="list-style-type: none"> • Lab help with emotional support • Bringing CS presence, motivation, inspiration and excitement into school • Relationships with the partner teachers and parents

Data from Middle School Students

Students' Pre-and Post- Content Assessments. Content assessments were scored for correctness. Each correct answer received one point while each incorrect answer received no points. The data was entered into an Excel spreadsheet and exported to SPSS for statistical analysis. A total score was calculated for the instrument as a whole for both the pre- and post- administration. Further, a T-test was performed to determine if there was a statistically significant difference between pre- and post- content assessment. The percentage of students who scored correctly on each item was calculated for both the pre- and post- administration of the assessment. The presence and absence of a variety of computational concepts was also analyzed.

Students' Pre- and Post- Design Tasks. Students were asked to save their Scratch projects with their initials, share and upload their projects to an online Scratch studio right after the end of each design task in the first and last sessions of the club. The uploaded Scratch projects were then saved to a local drive. All the programs were paired and matched. The programs were coded and analyzed using an online Scratch project-scoring website called "Dr. Scratch" (Moreno-León & Robles, 2015), which examines the presence and absence of computational concepts in students' programs (<http://www.drscratch.org>).

Students' Views of Scratch from Design Journals. The design and reflective journals students used during the computer club were first collected at the end of the club and scanned into digital versions. This data was analyzed qualitatively to identify emergent themes using the constant comparative method (Hatch, 2002). The author read all students' open-ended responses focusing on how they interpreted and

translated personal meaning of and relationship with Scratch and their attitudes toward Scratch.

Chapter 3

FINDINGS

The purpose of this study is to investigate the impact of a school-university partnership model and its associated field experience course activities on middle school students' CT development. It also aims to gain a better understanding of the ways in which college CS undergraduates supported the practicing teacher in the field, both through teaching practices and material development. In this chapter, the findings of each research question are presented and discussed.

Research Question 1

What specific roles did CS undergraduates assume in their partner school?

CS undergraduates assumed different roles in their partner school, including: (a) Technology Consultants; (b) Facilitators and Co-Teachers; and (c) Mentors and Ambassadors (see Table 11).

Table 11 Roles of CS Undergraduates in Partnerships

Roles of CS Undergraduates	Instances in The Field
Technology Consultants	<ul style="list-style-type: none">• Technical problem solving• Consultation on content skill; consultation on CS tools and recourses
Facilitators and Co-Teachers	<ul style="list-style-type: none">• Teaching and preparation: creating, planning lesson plan materials and delivering• Lab facilitator• Hands-on support

Mentors and Ambassadors

- Bringing connections to out-of-school identity as CS experts
 - Modeling ideal peer interaction
 - Gender equity
 - Near-peer mentoring as role models
 - Encouraging parental involvement and excitement
-

Technology Consultants. CS undergraduates were provided with rich CS content skills and resources to engage them as technology consultants. In the field, CS undergraduates were able to provide technology consultation on tools and technologies available for the target students. Further, CS undergraduates helped the teacher adapt curriculum materials available online and locate resources on given topics. Finally, they were also able to create sample Scratch programs needed to teach their lessons and solve a range of technical problems (e.g., install *Flash* on computers, trouble issues with Scratch accounts, etc.).

Facilitators and Co-Teachers. A key role in the field was that of a facilitator or co-teacher. Specifically, CS undergraduates frequently assisted the teacher with hands-on support and provided one-on-one student assistance. Analysis of blog entries, observation notes and interviews indicated that Ms. Sharon was grateful for that because it allowed her to provide students with timely support. In her interview, she noted: *“It is great to have extra pairs of hands... I couldn’t run this club without having them here to support the students.”* Further, all of the six undergraduates co-planned and co-led the lessons with Ms. Sharon. Lauren, one of the female CS undergraduates explained: *“Sharon was very clear with what she wanted us to do in her classroom and what she expected of us.”* Finally, in some instances, CS undergraduates were taking full teaching responsibilities. Mary explained, *“Ms.*

Sharon decided to pass along the reins to the UD students so from here on out, we will be taking charge in planning the objectives and activities for the day and teaching and controlling the classroom.”

Mentors and Ambassadors. One of the characteristics of this partnership model focused on the concept of students’ social and cultural learning environment. The presence of the undergraduates in the Scratch club provided role models for K-12 students that could help increase students’ engagement with CS. This near-peer mentoring offered access to CS identities and better understanding of what it means to be a computer scientist.

Research Question 2

How was the quality of the instructional materials created by CS undergraduates in the context of the school-university? How were CT concepts and practices implemented?

Quality of Instructional Materials

Results from the scoring of CS case reports are shown in Table 12. The findings, overall, indicated that lessons developed and delivered by undergraduates received high scores on each criterion and for the rubric as a whole ($M=3.6$, $S=0.62$). Overall, CS undergraduates actively sequenced, facilitated and co-led a set of progressive and cohesive CS lessons and activities for the Scratch club with the lead teacher in the field. They also collaborated as a team and communicated professionally with the teacher, students and faculty. In addition, CS undergraduates self-reflected on their field experience and team preparation. Details for each criterion are presented below.

Table 12 Average Mean Scores on the CS Case Report Rubric

Criteria	Mean	SD
Standard-Based Learning Objectives and Meaningful Topics	3.58	0.65
CT Practice-Infused Environments and Instructional Strategies	3.50	0.72
Age-Appropriate CT Technology and Activity Selection	3.71	0.69
Formative and Summative Assessment	3.62	0.71
Total	3.60	0.62

Standard-Based Learning Objectives and Meaningful Topics. Analysis of case reports indicated that most of the lessons’ learning objectives were clear. CT concepts or vocabulary were clearly and correctly presented and involved at least one of the big ideas in computing. Beth, for instance, explained in her journal, “*While we were waiting on tech support, I had a short conversation with the students in which I talked about computer science and some of its global impacts as well as what I did and why I studied computer science.*”

Topic selections were meaningful, appropriate and relevant to students and connected to prior knowledge or lessons (see Table 13).

Table 13 Description of Cases Enacted by Undergraduates: A Focus on Big Ideas

CS Principles Seven Big Idea Focus	Associated Activities from Case Reports	Number of Instances
1. Creativity	<ul style="list-style-type: none"> • Storyboard: Fall-theme projects • Recreating scenes from a book • Adding difficulty and extensions 	9
2. Abstraction		-
3. Data and Information	<ul style="list-style-type: none"> • Introduction to variables • Introduction to list 	3
4. Algorithms	<ul style="list-style-type: none"> • Boolean logic • Understanding how computer works (Marching Order) • Guessing game • Storyboard 	4
5. Programming	<ul style="list-style-type: none"> • Introduction to broadcasts • Introduction to loops • Introduction to conditionals • Advanced: variables and broadcasts combined • Interpreting code and programs • Recreating scene from school reading: Robin Hood Project 	9
6. The Internet		-
7. Global Impact	<ul style="list-style-type: none"> • Impact of CS 	1

The practicing teacher, Ms. Sharon also noted that working with CS students to identify modules, resources and lesson materials for the Scratch club gave her an opportunity to pilot ideas and look at what is effective, in addition to implementing a standard-based curriculum framework to her regular technology classes.

CT Practice-Infused Environments and Instructional Strategies. Findings indicated that CT practices and perspectives were encouraged and presented through undergraduates' teaching. The pedagogical methods and designed environments

included access to identity, peer interaction, modeling, scaffoldings or differentiated instruction (see Table 14). One CS undergraduate, Jason noted, “[students] were struggling a bit, so Mary and I went around and gave them help to get them up to pace.” (differentiated instruction). Mary indicated that some of the teaching practices were actually adapted after receiving input from Ms. Sharon during the debriefing meetings in the field. Mary noted, after one of the club sessions during their debriefs, “[Ms. Sharon] pointed out to us at the end of the class, we called on only boys as our volunteers, which she politely told us happens way too often and that the girls kind of get forgotten. Obviously, this is something I did unintentionally, but I will work on next time.”

Table 14 Description of Cases Enacted by Undergraduates: A Focus on CT-Infused Pedagogy

CT-Practice Encouraged Pedagogical Strategies	Number of Instances
CS Principles: Six CS Practices	
Connecting Computing	1
Creating Computational Artifacts	6
Abstracting	1
Analyzing Problems and Artifacts	2
Communicating	3
Collaborating	4
Equitable CS Instruction- Pedagogical Practices	
Strategically Partnering Students	2
Encouraging and Structuring Peer Interaction	8
Modeling Ideal Peer Collaboration	7
Differentiated Instruction	3
Kinesthetic Activity	8
Other	
Modeling or Demonstration	17
Scaffolding	1

Age-Appropriate CT Technology and Activity Selection. As shown in Table 15, the undergraduates selected a variety of CT tools or activities, such as Scratch projects or CS unplugged activities (see Figure 3), that were compatible and aligned with the learning goals and pedagogical practices needed to support student development of CT concepts and skills.

Table 15 Description of Cases Enacted by Undergraduates: A Focus on CT Tool Selection

Supportive CT Tools or Activity Selected	Associated Activities from Case Reports	Number of Instances
CS Unplugged Activity	20 Questions game (Boolean Logic) Logic puzzles (Boolean Logic) Marching order Loops Conditionals Variables Human computer interaction Introduction to pixel Skit	11
Scratch	Guessing game Pong game Moving sprite by user input Drawing shapes with Scratch Recreating scenes from a book Fall-theme projects Reading others' code Analyzing mystery programs Playing and discussing game features	10

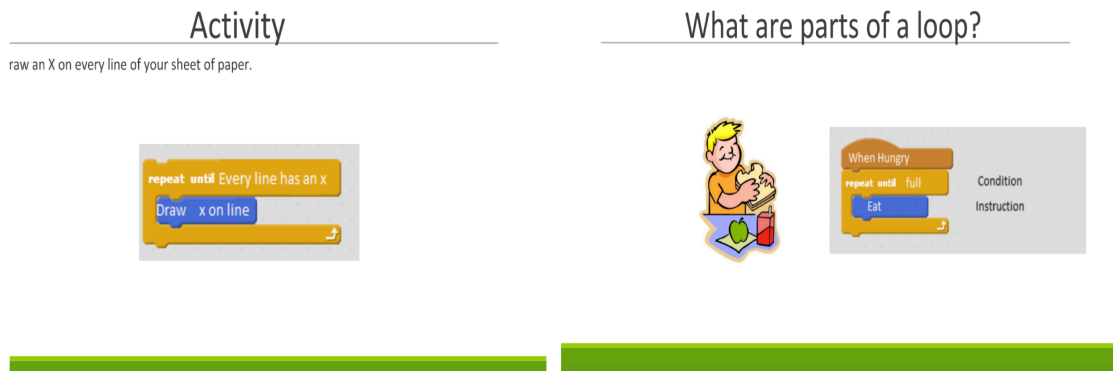


Figure 3 The Slides of CS Unplugged Activity: Loops

CS students also explained that some of the lesson activities were actually a result of discussing and collecting information from the teacher. Mary noted, *“We spoke with Ms. Sharon about what they're learning in school and we will try to incorporate some stuff they are learning into Scratch club for next week...Having this connection between school and Scratch is a good way to show how everything is related and could even show their English teachers their projects. This is another fun thing they can do with computer science besides just gaming.”*

Formative and Summative Assessment. Results also indicated that, overall, undergraduates were able to utilize different strategies (both formative and summative) to understand and assess students’ learnings. Formative assessment included exit tickets (Figure 4) at end of each session in the form of quick quizzes. Further, the design journals (Figure 5) documented students’ learning progress and project ideas (Figure 6) and served as a platform to document students’ understanding of CT knowledge and concepts (Table 16).

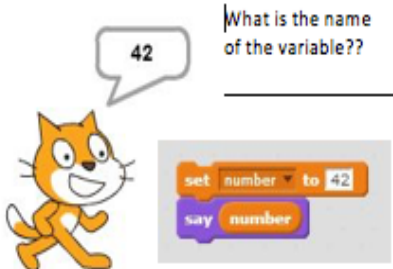
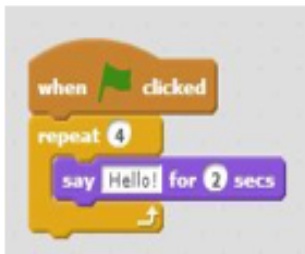
	
Variable Exit Tickets	Loop Exit Tickets
<p>Name : _____</p> <p>1. IF your first name starts with a vowel Draw a square ELSE Draw a circle</p>	
Conditional Exit Tickets	

Figure 4 The Exit Tickets Developed by Undergraduates

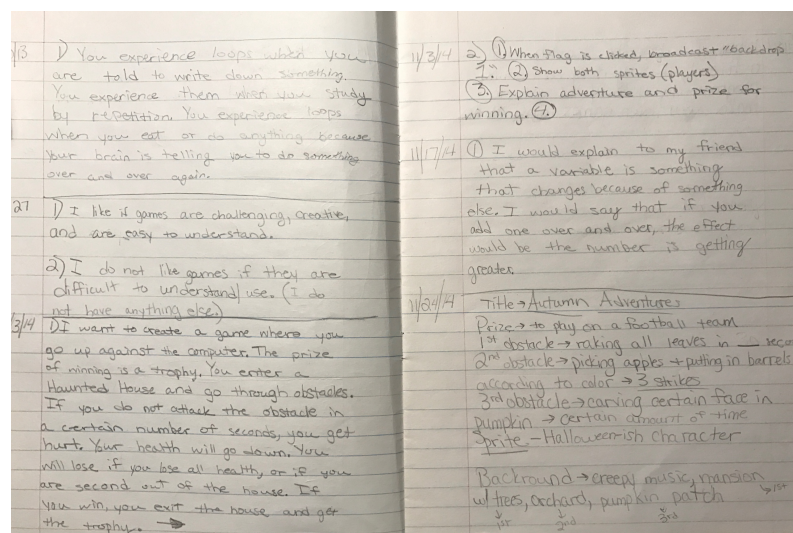


Figure 5 Students' Design Journal

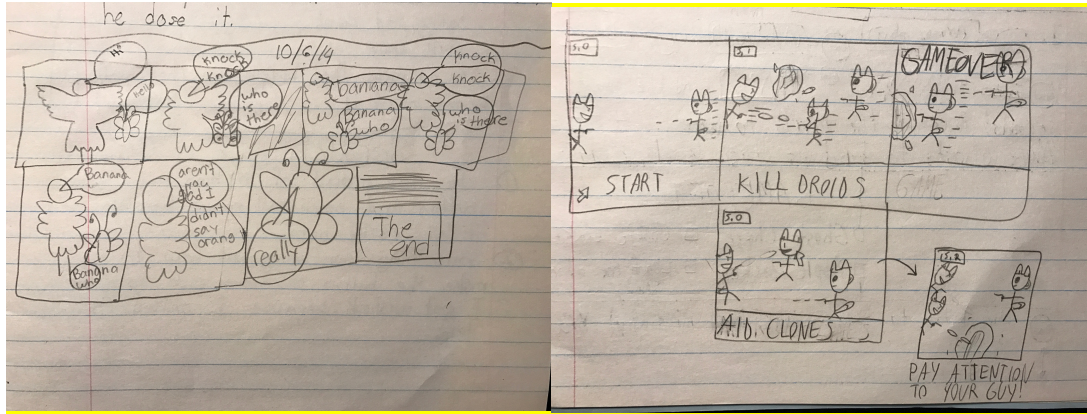


Figure 6 Students' Story Board

Table 16 Formative Assessment: Weekly Design and Reflective Journal

Sessions	Questions/ Prompts/ Tasks	Computational Thinking Assessed ⁴
Week 2	How would you describe Scratch to a friend? (Pre)	Expressing: realizing that computation is a medium of creation. Connecting: recognizing the power of creating with and for others.
Week 3	Storyboarding: Create a storyboard of their project.	Automating solutions through algorithmic thinking (a series of ordered steps)
Week 4 Week 5 (Continued)	1. List three ways you experience loops in real life. (e.g., going to sleep every night). For Advanced Students: 2. What are different ways of increasing difficulty in a game? 3. Which extensions did you add to your game project?	Loops: Automating solutions through algorithmic thinking (a series of ordered steps).

⁴ Brennan and Resnick (2012)

Week 6	1.What is something that works well or you really like about the project? 2. What are you most proud of? Why? 3. What is something that is confusing? 4.How did you get unstuck?	Identifying, analyzing, and implementing possible solutions with the goal of achieving the most efficient and effective combination of steps and resources.
Week 7	1. Describe the final project you want to create. 2. List the steps needed in order to create your project. 3. What might you need help with in order to make progress? 4. Editing Storyboard.	Experimenting and iterating: developing a little bit, then trying it out, then developing more. Testing and debugging: making sure things work and finding and solving problems when they arise.
Week 8	How would you describe “variables” to your friends?	Variables(data)
Week 9	Design a Fall-theme Scratch project on your journal and share your ideas.	The ability to communicate and work with others to achieve a common goal or solution.
Week 10	How would you describe Scratch to a friend? (Post)	Expressing: realizing that computation is a medium of creation. Connecting: recognizing the power of creating with and for others.

Summative assessments included several mini Scratch projects corresponding to the learning objectives of each session, and a final project which was shared publicly with parents on a demo day, where students presented their project creations and ideas (see Table 17).

Table 17 Description of Cases Enacted by Undergraduates: A Focus on Assessments

Formative or Summative Assessment	Associated Activities from Case Reports	Number of Instances
Exit Tickets	Broadcasts exit tickets Conditionals exit tickets Loops exit tickets Variables tickets	4
Design Journals with prompt Questions		7
Story Board	Fall-theme projects	2
Scratch Projects	Guessing game Pong game Moving sprite by user input Drawing shapes with Scratch Recreating scenes from a book	5
Final Demo (Show and Tell)	Fall-theme projects Projects of their own selection	3
Analyzing Codes	Reading others code and analyzing mystery programs Write down their guess and ideas	1
Analyzing Games	Playing with games and discuss the feature and design of each game	1

The performance-based assessment also included a showcase event at the University of Delaware. A group of students from the club at TCS were invited to showcase their Scratch games and projects to the university community members. Sharon noted *“This was huge to them. Till now some students still came to me said when they can go to UD again...”* These younger participants from TCS also got a chance during this event to try out projects and educational games that University of Delaware students created (see Figure 7).



Figure 7 A Group of TCS Students Invited to Participate in the CIS Showcase Event at UD

Research Question 3

What computational concepts did middle school students acquire as a result of working with CS undergraduates and their teacher?

Computational Thinking Acquisition in Students' Content Assessment

As shown on Table 18, results from the Scratch knowledge assessment indicated there was a statistically significant improvement ($p < 0.05$) from the pre administration of the assessment ($M=5.69$, $SD=2.09$) to the post administration of the assessment ($M=6.72$, $SD=2.18$) of the instrument as a whole with a medium effect size of 0.48.

Table 18 Dependent (Paired) T-Test of Pre- and Post- Content Assessment

Pre-Knowledge		Post-Knowledge		Mean Differences	t	df	P value Significance	Effect Size(d)
Mean	SD	Mean	SD	(Post – Pre)				
5.69	2.09	6.72	1.02	1.15	3.93	64	.000**	0.48

N=65, ** p<.01, * p<.05

Effect size <0.3 is small, 0.3-0.5 is medium, and >0.5 is large (Cohen, 1988)

Further, as shown on Table 19, results from the content assessment indicated improvements from pre to post administration on 9 out of the 10 questions (questions 1, 3, 4, 5, 6, 7, 8, 9 and 10).

Table 19 Pre- and Post- Scores on Scratch Knowledge Assessment Items

Questions and CS Concepts	Percent Correct	
	Pre	Post
1. Scratch block	92%	97%
2. Loop (repeated execution)	80%	72%
3. Handling an event	25%	40%
4. Data (modifying a variable)	46%	65%
5. Parallelism (broadcast)	68%	72%
6. Conditionals (if)	26%	45%
7. Loop (forever)	74%	86%
8. Loop (repeat a set number)	72%	78%
9. Conditional test	68%	85%
10.Script execution	23%	36%

Computational Thinking Concept Development in Students' Design Tasks

To gain a better understanding of students' CS learning, the investigator analyzed 23 pairs of Scratch projects. Analysis of students' pre-and post- design tasks is shown in Figure 8. In students' pre-design tasks, 65% of the programs were at the "Basic" level, while 35% were at the "Developing" level and none of them were at the "Master" level. However, compared to their initial design tasks, in the students' post design tasks, 57% were at the basic level, 39% were at the developing level, and 4% were at the Master level.

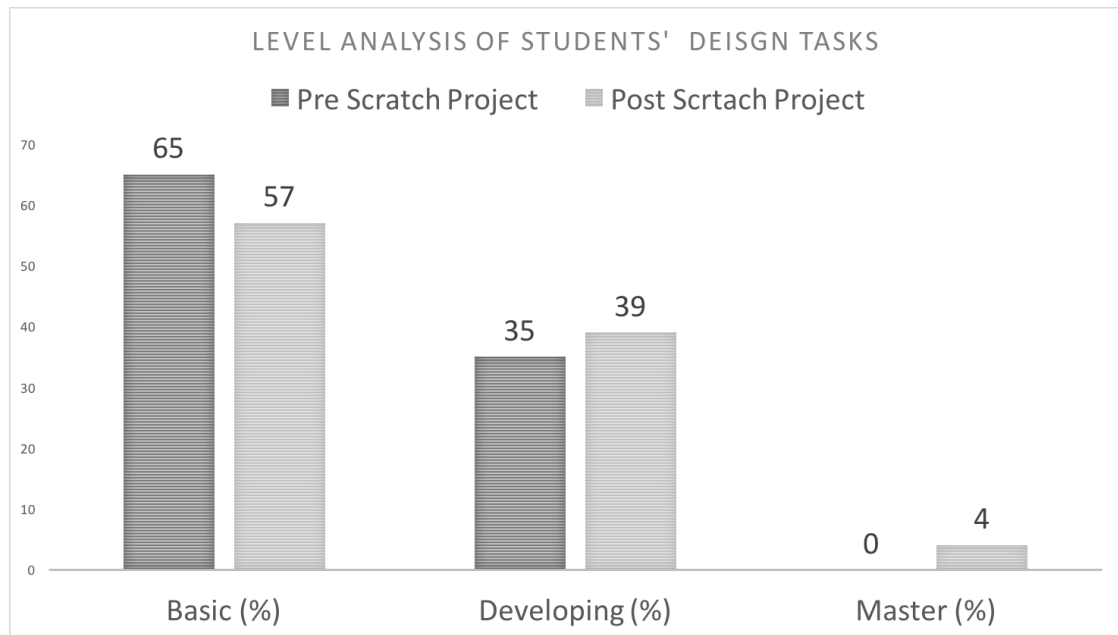


Figure 8 Scratch Project Level Analysis of Students' Pre-and Post- Design Tasks

Students' Views of Scratch

In the design journals, students were asked "*how do you describe Scratch to a friend?*" As shown in Table 20, the answers varied but can be categorized into the

following four emerging dimensions, including (a) Learning and Sharing Community (N=23), (b) Creativity and Fun (N=17), (c) Creating to Express (N=24), and (d) Scratch Features as a Programming Language (N=27). Students' responses captured the features of the Scratch and reflected on the purpose of the designed Scratch club learning environment presented in research question 2.

Overall, by analyzing pre to post responses, students were moving from computational consumers to creators. In the responses to the pre-question, students generally indicated that they could play and look at other people's projects as consumers (Dimensions A and B). In contrast, Dimensions C and D in the post-question received a slightly higher number of students' responses where they were able to identify the numerous possibilities of Scratch as a coding language to create anything imaginable to express themselves.

Table 20 Coding Schemes for Students' Views of Scratch

Parent Code	Initial Code	Entries Frequency and Percentages	
		Pre Responses	Post Responses
Learning and Sharing Community	• Community of people making fun games, animation		
	• Interaction with online games (Search and play)		
	• Share with community		
	• Programming club	15(26%)	8 (24%)
	• Visit it online or download it to computer		
	• Create games for others to play		
	• Remix other projects		
	• Programming/coding website		

Creativity and Fun	• Awesome, exciting, fun and interesting		
	• Creative		
	• Change color and sound, backgrounds	11(19%)	6 (18%)
	• Exciting and fun in a challenging way		
Creating to Express	• Put background and drawing		
	• Create video games, and stories, pictures, almost anything imaginable		
	• Make a lot of cool stuffs	14 (25%)	10 (29%)
	• Share and show your programming skills		
Scratch Features as a Programming Language	• Programming and code system		
	• Help students/adults learn programming		
	• Computer language for beginner and kids		
	• Use already-built blocks rather than typing		
	• Making project using characters	17 (30%)	10 (29%)
	• Cats runs around and execute command		
	• Drag and drop into another box		
	• Make/program objects to move		
	• Make projects with variables and scripts		

Learning and Sharing Community. In students' responses, some participants indicated that Scratch is a community of people making games. One student noted, Scratch is... *"a community of people doing it and it has fun game."* Another student responded, *"a fun project where everyone is making cool stuff. You could download it onto your computer. You could visit it online at scratch.mit.edu."*

This category is related to one element of CT perspectives, *Connecting*, identified and observed by Brennan and Resnick (2012). One student said, *"It's fun website to go on because you can make your own stuff and look at others."* Another student also explained, *"You can also play and search another person's game or project and play it."* These data indicate that students began to recognize "the power

of creating with and for others” (Brennan & Resnick, 2012). Students also realized that they could get inspired from other people’s work or build on existing projects or ideas when they are “*Remixing and collaborating.*” One student added, “*They [users] could remix the games or start from scratch.*”

Creativity and Fun. A number of students described Scratch as creative and fun. One student said, Scratch is “*a fun, creative and awesome way to show everyone your computer programming skills. SCRATCH is fun and creative.*” One of the seven big ideas in *Computer Science Principles* is *Creativity*, which suggests that creative development can be an essential process of creating computational artifacts.

Creating to Express. Most students reported that they would describe Scratch as a website where they can create games, stories and animations. One student noted, “*Scratch is a coding website. You can make games and pictures.*” Another student added, “*Scratch is a programming that you learn to create games, stories, and many more fun things.*”

Similarly, one element of CT perspectives identified by Brennan and Resnick (2012) is “*expressing*”, which argues that computation is a medium of creation which can be used for design and self-expression. Reflecting this idea, one student noted, Scratch is where “*you can show people your computer programming skills*” while another student added, “*Scratch is a website where you can program games animations and almost anything imaginable.*” Overall, students were able to understand that they can use Scratch to express themselves by creating computational artifacts.

Scratch Features as a Programming Language. Few students described Scratch with its blocks and low-floor, high-ceiling features. One student explained,

Scratch *“is a computer language used for kids who just began trying to program. You can use different tools to make projects you can share with the community. There is blocks already made used to program games.”* A second student added, *“you drag commands out of a box into another box to make sprites(character)move,”* while a third student described Scratch *“as a little yellow language cat that runs around and execute commands”*. Similarly a fourth student described Scratch as *“a place where you can make all different games/projects with variables/scripts”* Finally, some students also encouraged their peers to learn coding when responding to this question. One student said, *“that Scratch is a program on the computer that will help students and adults learn programming. It is a fun learning experiences. Everyone should learn.”*

Chapter 4

DISCUSSION AND CONCLUSIONS

Discussion

The purpose of this study is three-fold. First, it aims to describe the specific roles served by CS undergraduates in the context of the Partner4CS field experience model. Second, it seeks to examine the quality of the instructional materials they prepared to support partner teachers and students. Third, it aims to link CS undergraduates' practices from the designed partnership model to in-class support for teachers and subsequent student outcomes.

Data were collected from case reports enacted by the CS undergraduates, interviews and journal entries. Data were also collected from middle school students participating in the after-school program designed and implemented by a middle school teacher and CS undergraduates as part of the partnership model. Results indicated that the partnership model was able to bring in both material resources and non-material resources (Shah et al., 2013) to influence CS teaching and students' learning. Providing access to rich content skills, resources and quality instruction were considered as material resources while building out-of-school identity, teacher cooperation and parental involvement are considered as non-material resources. One of the CS undergraduates, Joe explained the value of parental involvement and how they also reached out to the communities through the partnership:

I got the chance to talk with some of the parents, and all of them were extremely thankful for the work we have done. Most of them said how enthusiastic their kids were about this stuff and how they would have never had the opportunity to learn these concepts through Scratch at such a young age. It made me feel good to hear that, and even though it's just an after-school club, hopefully this will get teachers and parents

to start putting pressure on the school boards to start making changes to the curriculum.

The analysis of case reports with associated reflection artifacts and investigator's observational notes suggested that Scratch lessons enacted by participants introduced meaningful and relevant CS concepts by utilizing sound pedagogical strategies and age-appropriate CT tools and activities. These lessons encouraged CT practices and perspectives among students' learning. Students' learning progress was also guided and built upon several formative and project-based assessments.

Similarly, students' views of Scratch provided evidence that designed lessons effectively engaged middle school students with CS content. The learning environment successfully encouraged students to employ many CT practices and grow as computational thinkers. Specifically, participation in the after-school program provided students with opportunities to acquire fundamental CS skills and transition from technology consumers to creators of computing innovations (Repenning et al., 2015). Further, the findings from data collected at the beginning and end of club in students' design tasks and content assessments indicated that students gained a better understanding of programming concepts and skills after their participation in the after-school program.

Recommendations

The recommendations are informed by the data sources described above and findings reported in literature. These recommendations are intently directed to the course instructors for the university field experience course and the Partner4CS team.

Scaffoldings on Lesson Plans and Teaching Strategies. Although CS students were provided with opportunities to independently experiment with the basic technology and to identify and adapt available CS curricula resources for engaging youth in CS, these students acknowledged that they still need help with lesson planning and teaching strategies. Students' reflections indicated that they had difficulty providing differentiated instruction. Mark noted, "*We also continued to have trouble getting all of the students to focus on their own projects. We'll have to come up with ways of keeping them more focused for next week... It's proving difficult to find a balance between challenging the less advanced students and maintaining the interest of the more advanced ones.*" Similarly, Beth who worked with Mark for a semester, also noted, "*how to challenge the more advanced students while not losing anyone else.*"

Further, data from participants' journals indicated that students appreciated teaching mock lessons, focusing on CS unplugged activities, Scratch and lab technologies, which provided them with opportunities to plan, design, rehearse and lead a lesson in front of their classmates for the first time. During these mock-teachings, they learned teaching practices from their peers and practiced how to give and receive feedback. However, in the future, course instructors could provide additional scaffoldings to address other related pedagogical issues. Students need additional supports related to learning theories, design of effective learning environments, problem solving and classroom management. Further, they found lesson planning a rewarding and valuable experience, but they acknowledged that it requires a significant amount of time to develop and adapt lessons and that they were not prepared for writing lesson plans.

Pairing Educational Students with CS Students. Course instructors could also consider recruiting more students majoring in education or mathematics or science education. The team could also identify potential opportunities or create a certificate to provide CS students with opportunities to co-plan and co-teach with education students in the field. The partnership model with an addition of education students holds promise for the successful infusion of needed pedagogies and lesson preparation for CS students, while simultaneously providing numerous benefits to education undergraduates in terms of CS content and skills.

Implications and Future Research

Findings of this work provided insights related to the impact and benefits of a partnership model to help middle students learn CS concepts. Future research could examine the benefits of the partnership model in different settings with diverse contexts. This study focused on upper elementary to middle-school students in the context of one after-school program. Future research could also investigate such partnership model and experiences within high schools, regular CS classrooms or informal setting. The contexts should include a more diverse sample in terms of gender, SES and ethnicity. In addition, future research should also include and employ a longitudinal study and larger scale of data collection across multiple years and locations. Relatedly, researchers can consider utilizing and developing other instruments, summative and formative performance-based assessments to collect and examine school students' CT development.

Further, future research may also investigate how the partnership model provides CS students with opportunities to hone their soft skills, such as, communication and leadership skills, and increase their confidence and technical

skills. Similarly, it would be beneficial to examine the partner teachers' knowledge and practices in teaching computing after their partnership experience and follow-up classroom support.

Findings from this work could also help establish best practices with regard to teacher education programs with the potential to support future pre-service CS teachers in practicum or internship opportunities. As needs grow in teaching computing in K-12 settings, many higher education institutions have started to redesign and develop graduate certificate programs for preparing K-12 CS teachers. The program curriculum can be divided into four areas that resemble aspects of the field experience course implemented in this work: (a) general studies: computing education in K-12, CS education curriculum resources and standards, and trending in teaching computing; (b) methods and assessment: pedagogical strategies and assessment; (c) CS technology knowledge and skills; and (d) practicum in computing education: much like student teaching, students are assigned to a semester-long field placement.

Limitations

The primary limitation of this study was its small number of participants. The data for the reflective journals and case reports were collected from six undergraduate participants. A larger sample size could have provided more insights. Also, the Scratch projects, content assessments and design journals were collected from 23 middle school students. Therefore, the small sample size limited the generalizability of the findings to the population. The investigation was also limited due to the short time frame and single location for data collection. A follow-up to this investigation could observe and collect more robust data over multiple years and across different

locations. A greater number of observations and data would lead to more significant and substantive findings to draw stronger conclusions.

Conclusion

Many new policy initiatives suggest that students need to be equipped with a better CS foundational knowledge and skills in order to understand the world around them, solve problems and become effective citizens in this computing-driven world (PCAST, 2010). The field experience course and its established partnership aimed at building a community of partners working together to broaden participation in CS education. The research-practice partnership model is attempting to find ways not only to train future STEM workers with awareness of community impacts, but to also seek new trends and instructional practices in teaching computing. This work presents an effective approach to the partnership model to broaden participation in computing among upper elementary and middle school students. The design and implementation of the partnership helped the school teacher to provide students with more opportunities to explore computational concepts and practices from an early age.

REFERENCES

- A is for Algorithm (2014). *The economist*. Retrieved from <http://www.economist.com/news/international/21601250-global-push-more-computer-science-classrooms-starting-bear-fruit>.
- Ahonen, A. K. & Kankaanranta, M. (2015). Introducing assessment tools for 21st century skills in Finland. In P. Griffin & E. Care (Eds.), *Assessment and teaching of 21st century skills* (pp. 213–225). Springer.
- Barr, D., Harrison, J., & Conery, L. (March 01, 2011). Computational Thinking: A Digital Age Skill for Everyone. *Learning & Leading with Technology*, 38, 6, 20-23.
- Barr, V., & 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.
- Berland, M., & Lee, V. R. (2011). Collaborative strategic board games as a site for distributed computational thinking. *International Journal of Game-Based Learning (IJGBL)*, 1(2), 65–81.
- Brady, C., Orton, K., Weintrop, D., Anton, G., Rodriguez, S., & Wilenski, U. (2017). All roads lead to computing: Making, participatory simulations, and social computing as pathways to computer science. *IEEE Transactions on Education*, 60(1), 59-66.
- Bransford, J. D., Brown, A., & Cocking, R. (2000). *How people learn: Mind, brain, experience and school* (Expanded ed.). Washington, DC: National Academy.
- Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. *Annual meeting of the American Educational Research Association*, Vancouver, Canada.
- Bureau of Labor Statistics, US Department of Labor. (2010). *Occupational Outlook Handbook, 2010–11 Edition, Computer Scientists*.
- Csizmadia, A., Curzon, P., Dorling, M., Humphreys, S., Ng, T., Selby, C., & Woollard, J. (2015). *Computing at School*. Retrieved from <http://www.computingatschool.org.uk>.
- Cohen, J. (1988). *Statistical power analysis for the behavioral sciences* (2nd ed.). Hillsdale, NJ: Erlbaum.

- College Board. (2014). AP Computer Science Principles Draft Curriculum Framework. Retrieved from <https://advancesinap.collegeboard.org/stem/computer-science-principles>.
- Computational Thinking—Teacher Resources second edition*. (2017) Retrieved from https://csta.acm.org/Curriculum/sub/CurrFiles/472.11CTTeacherResources_2ed-SP-vF.pdf.
- Computer Science Teachers Association, & International Society for Technology in Education. (2011). *Computational Thinking: Leadership Toolkit* (1st ed..). Retrieved from <http://www.csta.acm.org/Curriculum/sub/CurrFiles/471.11CTLeadershipToolkit-SP-vF.pdf>.
- CSTA Association for Computing Machinery. (2013). *Bugs in the System: Computer Science Teacher Certification in the U.S.* New York: Computer Science Teachers Association.
- CSTA Teacher Certification Task Force. (2008). *Ensuring Exemplary Teaching in an Essential Discipline: Addressing the Crisis in Computer Science Teacher Certification*. New York: Computer Science Teachers Association.
- Cuny, J. (2012). Transforming high school computing: A call to action. *ACM Inroads*, 3(2), 32- 36
- Delaware’s Science, Technology, Engineering and Math (STEM) Council (2012). *Annual Report: The State of STEM Education in Delaware*
- Denner, J., & Werner, L. (2007). Computer programming in middle school: How pairs respond to challenges. *Journal of Educational Computing Research*, 37(2), 131–150.
- Ericson, B., & McKlin, T. (2012). Effective and sustainable computing summer camps. In *43th SIGCSE technical symposium on computer science education* (pp.290-294).
- Ericsson, K. A., & Simon, H. A. (1993). *Protocol analysis: Verbal reports as data*. Cambridge,MA: MIT Press.
- Goode, J. (2011 Summer). Exploring Computer Science: An Equity-Based Reform Program for 21st Century Computing Education. *Journal for Computing Teachers*.

- Grover, S., Pea, R., & Cooper, S. (2016). Factors influencing computer science learning in middle school. SIGCSE, March 02-05, Memphis, TN, USA.
- Hatch, J. A. (2002). *Doing qualitative research in education settings*. New York: Suny University Press.
- International Society for Technology in Education (2016). National educational technology standards for students. Retrieved from <http://www.iste.org>
- Kolodner, J. L. (2006). Case-Based Reasoning. In K. L. Sawyer (Ed.), *The Cambridge handbook of the learning sciences* (pp. 225-242). Cambridge: Cambridge University Press.
- Maloney, J., Kafai, Y., Resnick, M., & Rusk, N. (2008). Programming by choice: urban youth learning programming with scratch. In *39th SIGCSE technical symposium on computer science education* (pp. 367-371).
- Matias, J.N., Dasgupta, S., & Hill, B.M. (2016). Skill progression in Scratch revisited. *CHI*, May 07-12, 2016, San Jose, CA, USA.
- Moreno-León, J., & Robles, G. (2015). Analyze your Scratch projects with Dr. Scratch and assess your computational thinking skills. In *Proceedings of the 7th international Scratch conference (Scratch2015AMS)*. Amsterdam, Netherlands.
- Mouza, C., Marzocchi, A., Pan, Y., & Pollock, L. (2016) Development, Implementation and Outcomes of an Equitable Computer Science After-School Program: Findings from Middle-School Students. *Journal of Research on Technology in Education*, 48(2), 84-104.
- Mueller, J., Becket, D., Hennessey, E., & Shodiev, H. (2017). Assessing Computational Thinking Across the Curriculum. In Rich, P. & Hodges, C. (Eds.), *Emerging Research, Practice, and Policy on Computational Thinking skills* (pp. 251–268). Springer.
- National Research Council (NRC). (2010). *Report of a Workshop on The Scope and Nature of Computational Thinking*. The National Academies Press.
- National Science Foundation (2014). College Board launches new AP Computer Science Principles course. Retrieved from http://www.nsf.gov/news/news_summ.jsp?cntn_id=133571.

- Office of the Press Secretary (2016). FACT SHEET: President Obama announces computer science for all initiative. Retrieved from <https://www.whitehouse.gov/the-press-of-offi ce/2016/01/30/ fact-sheet-president-obama-announces-computer-science-all-initiative-0>.
- Papert, S. (1991). Situating constructionism. In I. Harel & S. Papert (Eds.), *Constructionism*. (pp. 1–11). Norwood, NJ: Ablex.
- PCAST. (2010). *Prepare and inspire: K–12 education in science, technology, engineering, and mathematics (STEM) for America’s future*. Washington, DC. Retrieved from <http://www.whitehouse.gov/sites/default/files/microsites/ostp/pcast-stemedreport.pdf>.
- Pollock, L., Mouza, C., Atlas, J., & Harvey, T. (2015). Field experience in teaching computer science: Course organization and reflections. Proceedings of Special Interest Group in Computer Science Education, March 4-7, Kansas City, MO.
- Repenning, A., Webb, D., and Ioannidou, A. (2010). Scalable game design and the development of a checklist for getting computational thinking into public schools. SIGCSE ’10. (March 2010), 265-269.
- Repenning, A., Webb, D.C., Koh, K.H., Nickerson, H., Miller, S.B., Brand, C., Horses, I.H., Basawapatna, A., Gluck, F., Grover, R., Gutierrez, K., & Repenning, N. (2015). Scalable game design: A strategy to bring systemic computer science education to schools through game design and simulation creation. *ACM Transactions on Computing Education*, 15 (2), 11.1-11.31.
- Sengupta, P., Kinnebrew, J.S., Basu, S., Biswas, G., & Clark, D. (2013). Integrating Computational Thinking with K-12 Science Education Using Agent-based Computation: A Theoretical Framework. *Education and Information Technologies*, 18 (2), 351-380.
- Shah, N., Lewis, C. M., Caires, R., Khan, N., Qureshi, A., Ehsanipour, D., & Gupta, N. (2013). Building equitable computer science classrooms: elements of a teaching approach. *Proceedings of the 44th ACM technical symposium on computer science education* (pp. 263-268).
- Shute, V., Sun, C., Asbell-Clarke, J. (2017). Demystifying computational thinking. *Educational Research Review*, 22, 142-158.

- Webb, D., Repenning, A., & Koh, K. H. (2012). Toward an emergent theory of broadening participation in computer science education. In Proceedings of the 43rd ACM technical symposium on computer science education, February 29–March 3, Raleigh, NC (pp.173–178). New York, NY: ACM.
- Whitehouse.gov (2016). Computer science for all. Retrieved from <https://www.whitehouse.gov/the-press-office/2016/01/30/weekly-address-giving-every-student-opportunity-learn-through-computer>.
- Wing, J.M. (2006). Computational Thinking. *Communications of the ACM*, vol. 49(3), 33-35.
- Wing, J. (2008). Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society*, 366, July 2008, 3717-3725.
- Yadav, A., Burkhart, D., Moix, D., Snow, E., Bandaru, P., & Clayborn, L. (2015). *Sowing the seeds: A landscape study on assessment in secondary computer science education*. New York, NY: Computer Science Teacher Association.
- Yager, R. (Ed.). (1995). *Constructivism and learning science*. Mahway, NJ: Lawrence Erlbaum Associates.

Appendix A

PRE- AND POST- CLUB QUESTIONNAIRE

A.	WHAT IS YOUR NAME?	
B.	PLEASE SELECT YOUR GENDER: (check)	<input type="checkbox"/> Boy <input type="checkbox"/> Girl
C.	WHAT GRADE ARE YOU IN? (circle)	3 rd 4 th 5 th 6 th 7 th 8 th 9 th
D.	How many <u>Scratch Clubs</u> have you taken before? (please check)	<input type="checkbox"/> None <input type="checkbox"/> 1 <input type="checkbox"/> 2 <input type="checkbox"/> 3 <input type="checkbox"/> 4
E.	How many hours have you spent with <u>Scratch</u>? (please check)	<input type="checkbox"/> Between 0 and 10 Hours <input type="checkbox"/> Between 10 and 100 Hours <input type="checkbox"/> More than 100 Hours

1) In what category is the  block?

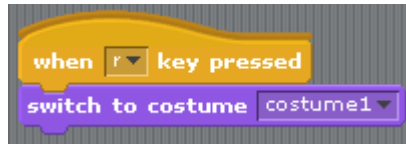
- a. Control
- b. Motion
- c. Sensing
- d. Variables
- e. Looks

2) What is the following an example of?



- a. Conditional execution
- b. Handling an event
- c. Sending a message
- d. Loop – repeated execution
- e. Modifying a variable

3) What is the following an example of?



- a. Conditional execution
- b. Handling an event
- c. Sending a message
- d. Loop – repeated execution
- e. Modifying a variable

4) What is the following an example of?



- a. Conditional execution
- b. Handling an event
- c. Sending a message
- d. Loop – repeated execution
- e. Modifying a variable

5) What is the following an example of?



- a. Conditional execution
- b. Handling an event
- c. Sending a message
- d. Loop – repeated execution
- e. Modifying a variable

6) What is the following an example of?



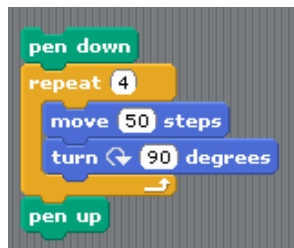
- a. Conditional execution
- b. Handling an event
- c. Sending a message
- d. Loop – repeated execution
- e. Modifying a variable

7) What does the following code do?



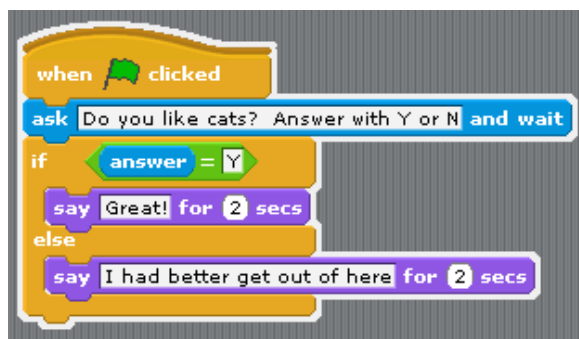
- a. Repeat a simple animation
- b. Draw a square using the pen
- c. Make a ball fall
- d. Increment the score
- e. Stamp the current costume at the current mouse location

8) What does the following code do?



- a. Repeat a simple animation
- b. Draw a square using the pen
- c. Make a ball fall
- d. Increment the score
- e. Stamp the current costume at the current mouse location

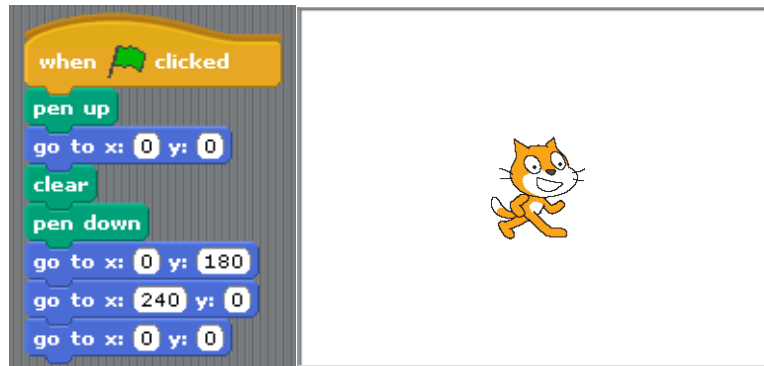
9) What will be said when the following executes and the user answers with No?



- a. Great!
- b. I had better get out of here

- c. I don't know
- d. It won't say anything
- e. You will get an error message

10) Draw the result of executing the following script when the cat is in the center of the stage.



Appendix B

IRB APPROVAL NOTIFICATION DOCUMENT



RESEARCH OFFICE

210 HULLIHEN HALL
UNIVERSITY OF DELAWARE
NEWARK, DELAWARE 19716-1551
Ph: 302/831-2136
Fax: 302/831-2828

DATE: April 27, 2016

TO: Chrystalla Mouza, Ed.D.
FROM: University of Delaware IRB

STUDY TITLE: [885606-1] Computational Thinking in K-12: Focus on Student Outcomes

SUBMISSION TYPE: New Project

ACTION: APPROVED
APPROVAL DATE: April 22, 2016
EXPIRATION DATE: April 21, 2017
REVIEW TYPE: Expedited Review

REVIEW CATEGORY: Expedited review category # (6,7)

Thank you for your submission of New Project materials for this research study. The University of Delaware IRB has APPROVED your submission. This approval is based on an appropriate risk/benefit ratio and a study design wherein the risks have been minimized. All research must be conducted in accordance with this approved submission.

This submission has received Expedited Review based on the applicable federal regulation.

Please remember that informed consent is a process beginning with a description of the study and insurance of participant understanding followed by a signed consent form. Informed consent must continue throughout the study via a dialogue between the researcher and research participant. Federal regulations require each participant receive a copy of the signed consent document.

Please note that any revision to previously approved materials must be approved by this office prior to initiation. Please use the appropriate revision forms for this procedure.

All SERIOUS and UNEXPECTED adverse events must be reported to this office. Please use the appropriate adverse event forms for this procedure. All sponsor reporting requirements should also be followed.

Please report all NON-COMPLIANCE issues or COMPLAINTS regarding this study to this office.

Please note that all research records must be retained for a minimum of three years.

Based on the risks, this project requires Continuing Review by this office on an annual basis. Please use the appropriate renewal forms for this procedure.

If you have any questions, please contact Nicole Farnese-McFarlane at (302) 831-1119 or nicolefm@udel.edu. Please include your study title and reference number in all correspondence with this office.