2022

KNITTING CODE: EXAMINING THE RELATIONSHIP BETWEEN KNITTING AND COMPUTATIONAL THINKING SKILLS USING THE NEXUS OF PRACTICE

Emily Rose Dodson-Snowden
edodson270@gmail.com
Author ORCID Identifier:
https://orcid.org/0000-0002-6791-8954

Digital Object Identifier: https://doi.org/10.13023/etd.2022.99

Recommended Citation
Dodson-Snowden, Emily Rose, "KNITTING CODE: EXAMINING THE RELATIONSHIP BETWEEN KNITTING AND COMPUTATIONAL THINKING SKILLS USING THE NEXUS OF PRACTICE" (2022). *Theses and Dissertations--Education Sciences*. 107.
https://uknowledge.uky.edu/edsc_etds/107

Emily Rose Dodson-Snowden, Student

Margaret Mohr-Schroeder, Major Professor

Dr. Molly Fisher, Director of Graduate Studies

KNITTING CODE: EXAMINING THE RELATIONSHIP BETWEEN KNITTING AND COMPUTATIONAL THINKING SKILLS USING THE NEXUS OF PRACTICE

DISSERTATION

A dissertation submitted in partial fulfillment of the
requirements for the degree of Doctor of Philosophy in the
College of Education
at the University of Kentucky

By
Emily Rose Dodson-Snowden
Lexington, Kentucky
Director: Dr. Margaret Mohr-Schroeder, Professor of STEM Education
Lexington, Kentucky
2022

https://orcid.org/ 0000-0002-6791-8954

ABSTRACT OF DISSERTATION

KNITTING CODE: EXAMINING THE RELATIONSHIP BETWEEN KNITTING AND COMPUTATIONAL THINKING SKILLS USING THE NEXUS OF PRACTICE

Due to the rise of careers in STEM-related fields, there is a growing need for schools to produce people to fill these positions. One area of STEM that is growing is computer science/coding. Due to this demand, schools need to be intentional about exposing students to computer science/coding. There are a variety of new tools to introduce students to this field. One growing belief is that knitting can teach computer science/coding to students.

The goal of this study was to see if knitting can serve as an introduction to teach students computation skills. Kitting has historically been used to code information, and numerous statements have been made that knitting can teach computer coding. The rationale behind this thought is that both fields have similar components and can serve to make coding more accessible to a broader audience. Suppose students that generally would not identify with computer science/coding due to perceived social norms develop an interest in knitting. In that case, they could use what they learned as a foundation to develop an interest in computer coding. This is based on Scollon's Nexus of Practice (2001), which studies how practices are linked together. This theory believes that combining different practices makes a possible crossover from one practice to another. As a result, what may not have been accessible at first due to biases or identity, may become more accessible. This study will focus on whether knitting can teach students computational skills and change students' identity towards computer science/coding.

There is limited research on the relationship between knitting and coding. This case study attempted to determine if knitting could teach coding. The research was conducted during two three-week summer enrichment programs. Results revealed that teaching computer coding through knitting was comparable to traditional instruction. While not necessarily better, this shows that knitting can teach computation skills and improve identity. This could be important for encouraging students that would not typically study computer science/coding to enter the field.

KEYWORDS: STEM, STEAM, Computer Coding, Knitting, Nexus of Practice, SCRATCH

Emily Rose Dodson-Snowden

(Name of Student)

03/21/2022

Date

KNITTING CODE: EXAMINING THE RELATIONSHIP BETWEEN KNITTING AND COMPUTATIONAL THINKING SKILLS USING THE NEXUS OF PRACTICE

By
Emily Rose Dodson-Snowden

Margaret J. Mohr-Schroeder
Director of Dissertation

Molly Fisher
Director of Graduate Studies

03/21/2022
Date

DEDICATION

I would like to dedicate this dissertation to the most important people in my life. First, to my parents, Jim and Roseann, who provided both emotional and financial support during the past five years. Second, to my husband, Josh, who constantly reassured me that I could finish this journey when I felt like I could not.

ACKNOWLEDGMENTS

The completion of this dissertation would not have been possible without the help of my dissertation committee, school, students, and fellow teachers. First, to my advisor, Dr. Margaret Mohr-Schroeder, who provided constant encouragement, suggestions, and feedback. Without her guidance, I would not have been able to produce this finished product. Second, to Dr. Jerzy Jaromczyk, Dr. Molly Fisher, and Dr. Cindy Jong who agreed to be a part of my committee. To Dr. Jaromczyk who agreed to take me on for an independent study and without his help this study would not have been possible. To Dr. Jong who made me question my thinking with constructive feedback. To Dr. Fisher who provided quick answers from the beginning to the end of this process.

To my school, students, and fellow teachers for helping me complete my research. First, to my school which provided the location and opportunity to carry out my investigation. Second, to my students for participating in my study and agreeing to learn how to knit. Finally, to my fellow teachers, Katie, Mark, Alex, Erin, Samantha, Clay, and Judy who provide constant support and laughter throughout the school day.

TABLE OF CONTENTS

LIST OF TABLES

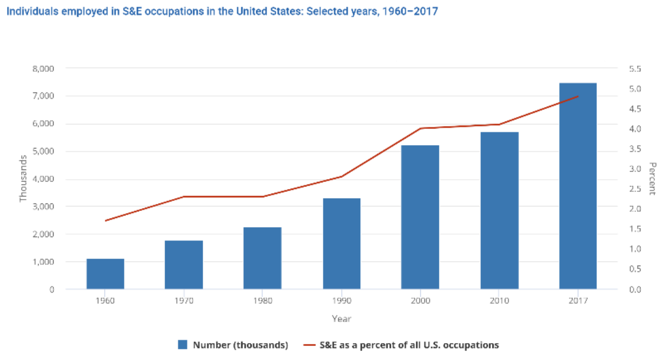
LIST OF FIGURES

Chapter 1 INTRODUCTION

With the rise in STEM (Science, Technology, Engineering, and Mathematics) careers there is a growing need for students to pursue majors in STEM fields to fill these positions. “In the 21st century, when the knowledge-based economy is steering improvement and development, STEM education has gained increasing momentum and importance” (Bozkurt, Ucar, Durak, & Iden, 2018, p. 374). The world is advancing, and STEM is at the center of this advancement. As a result, more people will need to fill the open positions. According to the National Science Foundation (2017), careers in science and engineering have increased from 1.7% in 1960 to 4.8% in 2017 (See Figure 1.1). The National Science Foundation (2017) defines science and engineering careers as biological/agricultural/environmental life sciences, physical sciences (chemistry, physics, astronomy, and earth/ ocean/atmospheric), computer sciences, mathematics/statistics, engineering, psychology, and social sciences. This does not include health care professionals or health care technicians.

Figure 1.1

Percentage of jobs that are science and engineering compared to total careers (National Science Foundation, 2017)



Science and engineering careers have more than double the growth rate of other jobs. Between 2019 to 2029, the US Bureau of Labor (2020) predicts that STEM careers will increase by 8.0%, while non-STEM careers will increase by 3.4%. Of these branches of science and engineering, mathematics and computer science are projected to make up 23.1% of the growth; additionally, of all the available science and engineering careers, 59% of them are predicted to be in math or computer science (National Science Foundation, 2017). Also, according to the National Science Foundation (2017), 51% of those who received a degree in mathematics, computer science, or engineering are most likely to get hired and maintain a career in their field.

One STEM field with a growing interest and needs is computer science (Massoud, Hallman, & Plaisent, 2018). With the rise of technology, more individuals that computer code will be necessary to keep up with the demand. Unfortunately, even though there is a rise in the need for computer coding, only 45% of high schools in the United States teach computer science (Code.org, n.d.). When students were asked which subjects were their favorite, they responded with computer science and the arts (Code.org, n.d.). Additionally, white males have been the majority to have plans to major in computer science in college (National Science Foundation, 2017). When students were exposed to advanced placement (AP) computer science courses in high school, females were ten times more likely to major in computer science in college, and Black and Latino students were seven times more likely (Code.org, n.d.). This shows that what occurs in a K-12 school before college can impact students' interests and career aspirations. Therefore, if more computer scientists are needed to fill the growing demand for computer coding careers, students need to be targeted before entering college.

While the demand has grown for computer coders, the computer science community has also adjusted and created new ways of teaching computer coding. One growing trend is called computer coding “unplugged”. Computer coding “unplugged” does not use computers but instead teaches computation skills using physical objects (Lee & Junoh, 2019). There are various ways that computer coding is being taught in “unplugged” formats, including drawing arrows on paper, playing board games, drawing out sequential events from a story, and using manipulatives (Lee & Junoh, 2019). Another tool that is being used to promote computer coding is codable toys. Various robotic toys have arisen that teach the essential components of computer coding, such as Lego Mindstorm, Ozobot, Sphero, Dash, and Dot. These robots require the user to create code and then transfer the code to the robot to perform certain functions. A third format in which kids are exposed to computer coding is through drag and drop programs. At the beginning of the century, Mitchel Resnick and Yasmin Kafai identified that a new method was needed to teach computer coding to children (Resnick, Maloney, Monroy-Hernandez, Rusk, Eastmond, Brennan, & Kaffai, 2009). This program became known as Scratch. Scratch is one of the first drag and drop programs. Drag and drop programs use colorful virtual boxes that look like long puzzle pieces and signify different possible commands. By fitting together these various commands, other actions occur. Resnick et al. (2009) thought Scratch/drag and drop would interest children due to the media creation, scaffolded approach, and colorful visuals.

Another alternative idea that is emerging to teach computer coding is through knitting. Knitting and computer coding have similarities, and as a result, there is a growing idea that students that learn to knit will be better computer coders (Roberts, 2019). In a New York Times article, Dr. Matsumoto argues that knitting is computer coding. She states that yarn is programmable, and by using different stitches in knitting (or crochet), a person can create different properties (Roberts, 2019). “Stitch patterns provide code…more complex code than the 1s and 0s of binary…that creates the program for the elasticity and geometry of knitted fabric.” The buzzword is “topological programmable materials,” said postdoc Michael Dimitriyev” (Roberts, 2019, Taking Yarn to the Big Screen Section, para. 2). By creating different properties, different outputs will be created, and that is how different designs are created. By teaching computer coding through knitting, the student is learning computer coding “unplugged”. Computer coding concepts are being presented but in a different format that may make the idea of computer coding more understandable and accessible (Roberts, 2019).

This case study focused on the idea of using knitting to teach computer coding. The purpose was to combine knitting instruction with computer coding instruction to observe the outcomes of student identity and development of computational thinking skills. While this may seem like an abstract idea, this is not the first-time computer coding, and knitting has been combined.

1.1 History of Knitting Code

Knitting has a history of being used as a coding mechanism. During WWI and WWII, female spies would code messages into knitting (Petersen & McClintock, 1942). During WWI, an elderly Belgian woman would knit as she looked out her window at the train station. As one train passed, she would make a purl in the fabric she was knitting. When another train passed, she would intentionally drop a stitch to make a hole (Petersen

& McClintock, 1942). By doing this, she could code the comings and goings of trains. She then gave this piece of knitted fabric to a fellow Belgian resistance spy who was working on defeating the occupying Germans (Petersen & McClintock, 1942). During WWII, a British woman named Phyllis Latour Doyle would sit outside a German War Office and knit messages for the British. She would incorporate Morse Code into her fabric, using knit and purls, as officers came and went from the war office (Fear, 2018).

Knitting slowly started to be known as a way to code information. At the end of WWI, an article appeared in UK Pearson's Magazine, stating Germans had knitted whole sweaters filled with messages (Adlington, 2015). The article continues that when the Germans unraveled the sweater, they discovered knots in the wool yarn. There were different spaces between each knot which represented different letters of the alphabet (Adlington, 2015). During WWII, Belgium placed a ban on posting knitting patterns because they may have contained code, and the British refused to take imported patterns in case they contained code (Adlington, 2015).

Knitting was very common during wars when resources were not as available, so it was not uncommon to see a woman with knitting needles and yarn. Women were often knitting socks or mittens for soldiers. This is what made knitting a perfect cover. Knitting looked innocent enough, but the information was gathered without anyone knowing (Adlington, 2015).

Knitting Code can also be seen in the literature. In the book "A Tale of Two Cities" by Charles Dickens (1859), Madame Defarge, a French woman, and worker of the French Revolution, would watch the executions of nobles while knitting. While the knitting looked inconspicuous, Madame Defarge was knitting names of future nobles she would condemn to the guillotine. Her knitting listed the names of everyone the French revolutionaries meant to kill (Dickens, 1859).

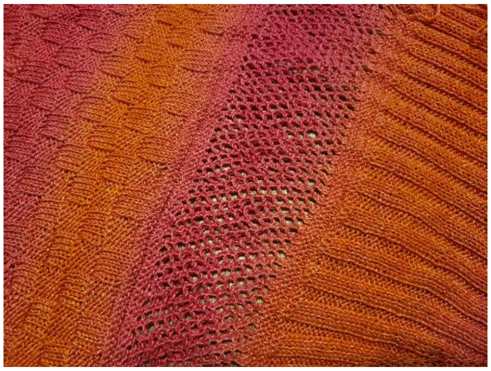
Knitting Code is a form of steganography. In steganography, messages are hidden in objects, such as hiding Morse code on a postcard or knitting messages in a scarf (Brandreth, 1899). Knitting is formed from two types of stitches, the purl, and the knit. The purl produces a small bump while the knit produces a "v" shape stitch. Different looks occur when the stitches are combined in different ways, but the knitting is still only made up of knits and purls. Knitting provides a good medium to write Morse code because knits and purls can represent dots and dashes (Brandreth, 1899). Even in today's modern society, knitting is still used as code. In Isabell Kraemer's (2019) "Purl Code" sweater, knitters can knit messages into their sweater through a Morse Code alphabet. To understand how this works, a basic understanding of knitting is needed.

1.2 Information on Knitting and Computer Coding

A basic knitting and computer coding background are necessary to understand how Knitting Code works. First, an understanding of knitting and reading knitting patterns is needed. In knitting, needles, and yarn create either a knit or purl stitch. The placement of the yarn and needle dictates the type of stitch made. Each type of stitch produces a stitch that varies in appearance. Additional steps can be placed between a knit or purl, such as moving the piece of yarn being used in something called a yarn over to create a more decorative or structurally sound stitch. Overall, the two stitches make a variety of looks, from lace to the typical knit sweater. A beginner knitter is taught both stitches, but the combinations of the stitches can produce hundreds of different looks. Examples of different stitch combinations can be seen in figure 1.2.

Figure 1.2

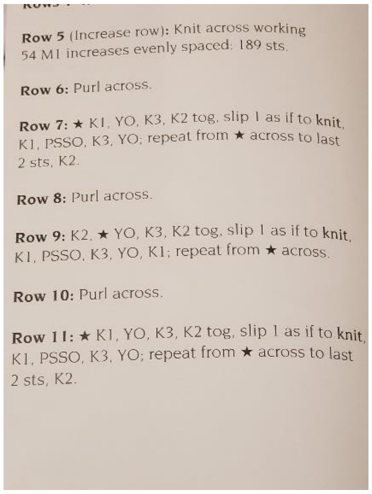
Sample of knitted work using the knit and purl stitches. The combination of the stitches produces different textures and appearances. Knitted work based on the pattern from Pendenza Sampler Shawl (Plymouth Yarn Design Studio, 2017)



When reading or writing a knit pattern, the goal is to simplify the writing enough so that the pattern uses repeats instead of writing the same directions repeatedly. These repeats are necessary. Otherwise, the pattern would become confusing and very long. Repeats are usually signified by using an asterisk, symbol, or brackets with an explanation of how many repetitions are necessary. Additionally, these repeats may vary due to the size of an item the knitter is trying to create. Abbreviations are also used in patterns so that the pattern writer does not need to write out each stitch. It is commonly known that a “K” represents a knit stitch while a “P” symbolizes a purl stitch. Additional abbreviations are usually listed in a key on the pattern, such as K2Tog (knit two together) or SSK (slip a stitch, slip a stitch, and knit both stitches). The overall purpose of abbreviations and repeats is to simplify the pattern to be easy to read and take up minimal space. An example of a written knit pattern can be seen in figure 1.3.

Figure 1.3

Example of knitting pattern instructions. Instructions show an example of abbreviations and repeats (stars). Instructions are from Knit Cowls by Lisa Gentry (2014)



Row 5 (Increase row)**:** Knit across working 54 M1 increases evenly spaced: 189 sts.

Row 6: Purl across.

Row 7: ★ K1, YO, K3, K2 tog, slip 1 as if to knit, K1, PSSO, K3, YO; repeat from ★ across to last 2 sts, K2.

Row 8: Purl across.

Row 9: K2, ★ YO, K3, K2 tog, slip 1 as if to knit, K1, PSSO, K3, YO, K1; repeat from ★ across.

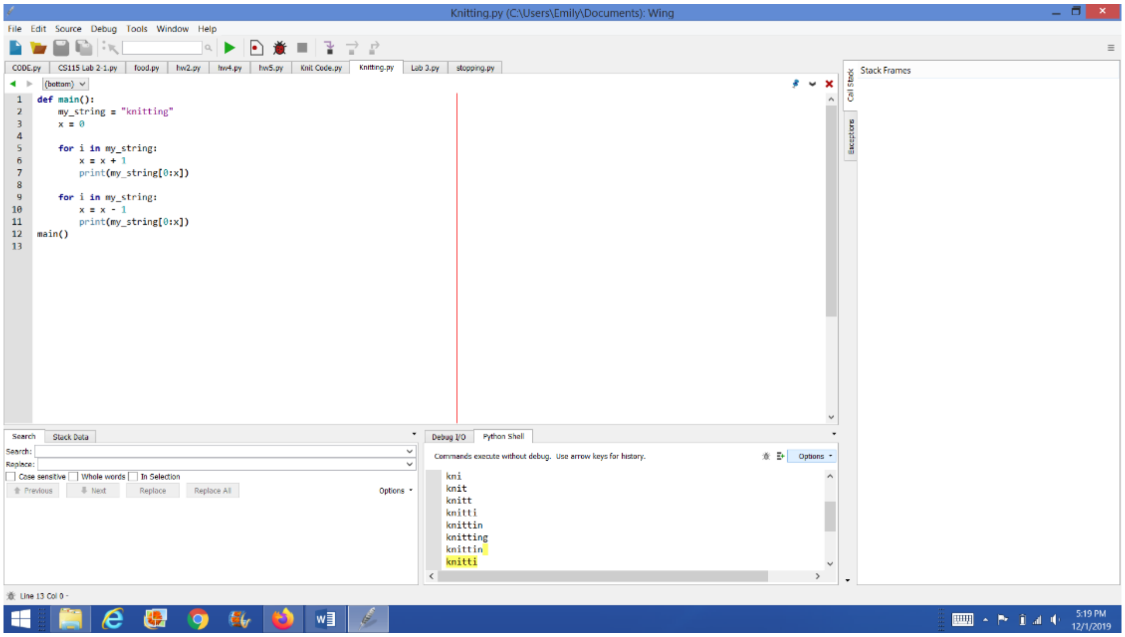
Row 10: Purl across.

Row 11: ★ K1, YO, K3, K2 tog, slip 1 as if to knit, K1, PSSO, K3, YO; repeat from ★ across to last 2 sts, K2.

As seen in the knitting pattern in Figure 1.3, knitting has its own type of coding, much like computer coding. While they are very similar, computer coding can sometimes be more complicated. First, there are a variety of languages that can be learned, from C++ to Python. Additionally, in computer coding, there is both input and output. In the input, something is placed in the code that changes the output. The computer coder also writes lines of codes that provide directions to produce the output. Variables are also used to assign data to a group, such as variable_one = 100. Functions are also used to direct data in a piece of written code. Finally, loops are used to create repeats until the desired outcome is reached. A loop is “a sequence of instructions that are continuously repeated until a certain condition is reached” (Lee & Junoh, 2019, p. 712). Looping is a “necessary process for effectively coding an event that occurs repeatedly” (Lee & Junoh, 2019, p. 712). Figure 1.4, code is written in Python and shows variables, input, output, and strings. Knitting and computer coding at first appear to be very different, but they have several similarities.

Figure 1.4

Example of code and output. The variable is my_string and represents “knitting.” The code is a loop and repeats until knitting is spelled out letter by letter, and then each letter is taken away.



```
def main():
    my_string = "knitting"
    x = 0

    for i in my_string:
        x = x + 1
        print(my_string[0:x])

    for i in my_string:
        x = x - 1
        print(my_string[0:x])
main()
```

Commands execute without debug. Use arrow keys for history.

```
kni
knit
knitt
knitti
knittin
knitting
knittin
knitti
```

1.3 Definitions

Knitting: The use of needles and yarn to produce a fiber product

Knit: A knitting stitch that forms a flat stitch

Purl: A knitting stitch that forms a small bump

Pattern: Directions of when to use knit and purls and how many

Parallelization: Acttions occurring at the same time (Merriam-Webster, 2022)

Identity: “The connections someone makes between themselves and a social group” (Starr, 2018, p. 490).

Computer Code: “The process of identifying and labeling each step to complete a task” (Lee & Junoh, 2019, p. 712)

Computer Language: Different styles of commands that used for carrying situations

Input: A command that is placed in the code that changes the output

Output: The end product

Variable: A placeholder

Loop: A repeat

Function: Used to direct data in a piece of written code

Sequence: The order of a set of steps

Algorithm: A list of step-by-step procedures used to complete a task (Lee & Junoh, 2019).

Decomposition: Problems are broken down through a process (Sung, 2019)/ the process of breaking down complex problems into smaller, more manageable parts (McVeigh-Murphey, 2019).

Abstraction: Ignoring specific details to focus on the general idea (Sung, 2019)/ to simplify strings of code into different functions (McVeigh-Murphey, 2019).

One issue that arose when referring to terminology was the use of computer coding. Coding is an ambiguous term that can be applied to several situations and is not specific to writing a computer language. The correct terminology for writing a computer language would be computer programming. While computer programming is what this study is encompassing the researcher used the vaguer word of computer coding due to the terminology Scratch uses. Scratch was the program the researched used in the study, and as a result students learned the word coding. When trying to add the word programming students because confused, so the researcher referred to their actions as coding. To streamline what was learned by students/student quotes, computer coding is used throughout this paper instead of computer programming. The word computer is added before coding to designate the application to computer science/programming compared to the coding mentioned in data analysis.

1.4 Similarities and Differences Between Knitting and Computer Coding

Computer coding and knitting have some similarities and differences. Computer code can be very complex, and while knitting can be tricky, it is only made up of two stitches. Computer code has its own languages with multiple words and meanings. One argument for the similarities between computer code and knitting is that knitting uses a knit and purl while computer coding uses 0's and 1's (Roberts, 2019). This would be true for binary, but this is not true for using most computer coding languages. Second, the knitting pattern and computer code produce an output, but the output varies. Computer coding produces words, directions, and images, while the knitted pattern produces a knitted fabric.

There are also some significant similarities between how computer coding and knitting are constructed. When reading the patterns, you read the directions in lines from left to right in either case. In both cases, the rules of reading the directions are the same, and after completing one line, the reader or computer completes the next step of orders. Second, both the computer code and knitting pattern have input and output. The input for knitting is yarn, and the output is a section of knitted material. In computer code, the input and output are text. There are repetitive sections in both a knitting pattern and computer coding. In computer coding, the repetitive areas are called loops and repeat until some function is completed. In knitting, the repeats continue until a certain number of stitches are attained. In both cases, once a marker is met, the computer or knitter knows to move to the next section. According to Buckner (2015), knitting and coding minimize repeating directions/code through a loop. A loop ends in either case once a specific pre-set parameter is met. For knitting, this might be at the end of a row. Additionally, in knitting instructions, meanings are assigned to certain abbreviations, similarly to the variables assigned in computer code.

Knitting and computer coding also have some additional similarities. First. Both knitting and computer code store information. A knitted item stores all the steps that occurred in the process. Additionally, data can be knitted into a pattern. For example, certain temperatures outside could be assigned a color. The knitter then could knit a color to represent the average temperature for each day for a year. In computer coding, the input information is stored and used in the computer code. Second, frequent error checks are needed in both knitting and computer coding. An error early on could destroy the computer code or knitted pattern later. Therefore, regular checks are needed in both to make sure the pattern and computer code are working. Third, in both knitting and computer coding, learning the initial ideas are simple, but both have large areas to improve and grow. Besides the similarities between knitting and computer coding, there are also different types of computer code that can be knitted.

1.5 Knitting is Code/Types of Knitting Code

There is a variety of formats that can be used to knit code. During WWI and WWI, spies would use different combinations of stitches and knots to send messages. In today's modern society, coding knitting has expanded. In Elizabeth Kraemer's (2019) Purl Code sweater pattern, knitters have the opportunity to knit a Morse Code message into their sweater. This is one way that code can be incorporated into knitting.

Another method of Knitting Code is by using binary code. In binary, the numbers 0 and 1 represent the on and off switches of what is occurring in a computer. This can be transferred to knitting by using knits and purls to represent the 0's and 1's (Roberts, 2019). A knitter can also use two yarn colors to represent the 0's and 1's. A third method is to knit the numbers/symbols/words/pictures into the pattern in something called typographic knitting.

Typographic knitting is “the process of fitting color fields together in a woven design” (Schlomer, 2019, p. 4). The combination of the different colors creates a pixel-like effect, which enables the knitter to create an image. What is unique about typographic knitting is that it “demands constant translation between the physical knitting process and the optical result” (Schlomer, 2019, p. 5). This pixel form of knitting requires the knitter to use logical and pre-assigned colors to create a design that portrays a pre-planned image. Pictures on computers are made of pixels that have been assigned a specific color. In both knitting and computer coding, a picture/pattern is created when these pixels are combined. Pixel images are a form of code, and this can be applied to knitting by either using knits and purls or using different colors of yarn. By creating pixel images on knitted materials, the designer can portray codes. For example, the knitter can knit 0’s and 1’s for binary code or knit in dashes and dots for Morse code. Additionally, the knitter can knit designs such as a QR code or a pixelized image.

Overall, there are several applicable ways the code can be portrayed through knitting. Additionally, many statements have been made that Knitting Code is something that teachers should be considering (Roberts, 2019). Knitting Code is a unique “unplugged” method of teaching code.

1.6 Research Questions

This study aims to better understand if knitting can improve students’ understanding of computer coding. Claims have been made that knitting is code (Roberts, 2019), there are various historical examples, and there are multiple forms of knitted code. In this study, two groups of students between sixth to eighth grade from a public middle school summer school enrichment program will either undergo a computer coding program (comparative group) or a knitting and computer coding program (experimental group). The program will take fourteen 48-minute class periods, with the control group being taught a computer coding curriculum. In contrast, the experimental group is being taught a computer code curriculum that incorporates knitting. By the end of the study, students in the comparative group will have a basic understanding of computer coding, and students in the experimental group will have a basic knowledge of computer coding and knitting.

It is believed that the use of arts in STEM can promote a different type of learning due to the requirement to find a creative solution to a problem (Peppler & Wohlwend, 2018). The key is that the art/craft must be included in a meaningful way that improves learning through problem-solving and not just to make an item. By using craft/knitting in combination with coding, the identities clash and produce a new learning experience that may enable females, minorities, or those not typically interested in computer coding to excel (Peppler & Wohlwend, 2018). This study will focus on two research questions.

- Research question 1: How does learning to knit while learning to computer code facilitate the understanding of Computational Thinking skills, including abstraction, Algorithmic Thinking, and understanding of parallelization in adolescent students?
- Research Question 2: How does combining knitting and computer coding impact the identity of who studies computer science in adolescent students?

These research questions were determined through a thorough investigation on literature related to STEM, computer coding, computer coding “unplugged”, minority/female identity, arts in STEM, and the Nexus Theory. Several themes emerged, including identity, learning skills, and teaching methods.

1.7 Overview of Study

The Nexus Theory is the guiding Theoretical Framework of this study. It is believed that the use of arts in STEM can promote a different type of learning due to the requirement to find a creative solution to a problem (Peppler & Wohlwend, 2018). By using craft/knitting in combination with computer coding instruction, the identities clash and produce a new learning experience that may enable females, minorities, or those not typically interested in computer coding to excel (Peppler & Wohlwend, 2018).

The research will be designed as a qualitative case study. Data will be collected in various formats to determine trends and eliminate bias. Data will be collected during every lesson through a variety of formats. At the end of the study, data will be coded using QDA Minter Lite to determine categories and themes. Finally, data from the comparative group will be compared against the experimental group to serve as a baseline.

Chapter 2 LITERATURE REVIEW/THEORETICAL FRAMEWORK

STEM careers are in very high demand (Bozkurt et al., 2018). "In the 21st century, when the knowledge-based economy is steering improvement and development, STEM education has gained increasing momentum and importance" (Bozkurt et al., 2018, p. 374). One component of STEM is technology, and with the increasing use of technology, computer science, and computer coding are becoming vital in today's society (Massoud et al., 2018). "It is getting more and more widely recognized that teaching children the basic concepts and skills of computer sciences are of considerable value. It seems that, among many technological applications such as software, hardware, and the internet, the software application has gained prominence in computer sciences in recent years" (Tonbuloglu & Tonbuloglu, 2019, p. 404).

Personal computers became available to the general public in the late 1970s. As a result, there was an increase in enthusiasm to teach children how to computer code (Resnick et al., 2009). This enthusiasm slowly faded with time, and schools shifted away from teaching computer coding to using computers for other purposes (Resnick et al., 2009). While computers are still an essential part of today's society, and children often use them daily, studying computer coding has become exclusive. As a result, computer coding has developed an identity that it is only attainable for a small group of technically and mathematically skilled people (Resnick et al., 2009). This is problematic because computer coding is becoming more important and needed as technology improves. There are not enough people to fill the demand (Massoud et al., 2018; Maxwell, 2016; National Science Foundation, 2017).

Computer coding is “the process of identifying and labeling each step to complete a task” (Lee & Junoh, 2019, p. 712). There is an increase in demand for those that know how to computer code, and as a result, colleges need to produce a diverse group of learners to fill this demand (Ehrlinger et al., 2018; Varma, 2006). Computer “coding skills should be considered among basic skills, and they are of equivalent importance as reading” (Tonbuloglu & Tonbuloglu, 2019, p. 404). Although, all responsibility should not be placed on colleges because students develop an interest in computer science at a much younger age (Sung, 2019). Additionally, students develop an identity as they grow and are exposed to computer science (Starr, 2018). If this identity does not match the stereotype of a computer scientist that the individual has formed, there will be a loss of interest (Starr, 2018; Varma, 2006).

Various tools have been created to develop an interest in computer science and computer coding for children. One tool is called computer coding “unplugged” and teaches computer coding skills without using technology (Lee & Junoh, 2019). Another method used to teach computer coding to children is using computer coding toys such as Lego Mindstorm, Ozobot, Sphero, Dash, and Dot (Gurbuz, Evlioglu, Erol, Gulsecen, & Gulsecen, 2016). Finally, drag-and-drop programs, such as Scratch, have been developed to make coding more scaffolded and engaging (Resnick et al., 2009). Another growing idea is that knitting can be used to teach computer coding. Knitting and coding have similarities, and as a result, there is a growing idea that students that learn to knit will be better coders (Roberts, 2019).

There have been numerous claims that knitting can teach computer coding concepts, but these claims have little scientific backing (Buckner, 2015; Roberts, 2019).

As a result, a literature review is needed to determine trends and previous/current research. From the literature, the best methods and tools can be selected to design an experiment to test the claim that knitting can teach computer coding. Since there is minimal research on teaching computer coding through knitting, or as the researcher has termed it, “Knitting Code”, it is impossible to search for articles related to this idea. Instead, the ideas surrounding “Knitting Code” had to be used to look at past/current research. Keywords and concepts used when searching for articles included: computer coding “unplugged”, computer coding, computer coding, computer science, STEM, and STEAM. Additionally, the terms Computational Thinking and Algorithmic Thinking were added as search terms when they kept appearing in articles. Finally, while reading, female involvement and identity issues in computer science arose. As a result, further searches included the key terms: females and minorities. As the literature was reviewed, themes started to appear to explain how knitting could become a future tool for teaching computer coding.

2.1 Literature Review

This literature review is organized into four main sections based on the four emerging themes while reading past/current research. The four themes are identity (Ehrlinger et al., 2018; Rubio et al., 2014; Starr, 2018; Tobin, Menon, Menon, Spatta, Hodges, & Perry, 2010; Toglia, 2013) essential thinking skills (Cooper et al., 2000; Futschek, 2006; Hurlburt, 2018; Lee & Junoh, 2019; Lee & Junoh, 2019; Ricketts, 2018; Sung, 2019; Tonbuloglu & Tonbuloglu, 2019), STEM vs. STEAM (Adkins et al., 2017; Gulhan & Sahin, 2018; Jawad et al., 2018; Looi et al., 2018; Peppler & Glosson, 2013; Thuneberg et al., 2017), and pedagogical teaching approaches (Adkins et al., 2017;

Campbell & Walsh, 2019; Costantino, 2018; Daugherty, 2013; Gulhan & Sahin, 2018; Lee & Junoh, 2019; Peppler & Glosson, 2013; Resnick et al., 2009). Each section will begin with a summary of the theme, followed by a more in-depth review of what is found in each theme, and will then end with an overview of research that encompasses that theme. Upon completing the Literature Review, the Theoretical Framework will be discussed. Finally, in a brief conclusion, the components that will be used in the "Knitting Code" program will be discussed based on the information from the Literature Review and Theoretical Framework.

2.1.1 Theme One: Identity and the Impacts

A person's identity impacts their interest, what major they choose in college, and what career they choose for their future (Starr, 2018). An identity is "the connections someone makes between themselves and a social group, such as people in STEM fields" (Starr, 2018, p. 490). Females and other minorities, such as Blacks, have been shown to struggle to align their identity with that of a computer scientist (Ehrlinger et al., 2018). Additionally, males and females have different perceptions of who computer codes (Rubio et al., 2014). When a person's perceived stereotypes about a group, in this case, computer scientists, do not match their own self-concepts or identity, their desire to be a member of that group diminishes (Tobin et al., 2010). Due to this disconnect between identity and females/minorities, there is an underrepresentation of these groups in computer science careers even though the demand is rising (Rubio, et al., 2014). While there are personal costs for those not pursuing these careers, such as less income, there is also a national and world cost. The denial of diverse perspectives that may bring a different perspective, and thus solution, to a problem, are not present (Ehrlinger et al., 2018). Rubio et al. (2014) state that male students claim to find computer coding easier, have higher intentions to program in the future, and show higher learning outcomes than female students.

Gender plays a role in learning and should be considered when designing lessons (Honigsfeld & Dunn, 2003). Males and females have distinct learning preferences such as environmental, emotional, sociological, physiological, and perceptual learning styles that should be considered when designing lessons (Honigsfeld & Dunn, 2003). For example, according to Honigsfeld and Dunn (2003), male students preferred peer interaction and kinesthetic activities, while females preferred warmer temperatures, various social interactions, self-motivation, persistence, and responsibility. If computer science is only taught using one method, then that method may not be appealing to females and have nothing to do with the content but with how the content is being presented. This may be able to explain why females could potentially struggle with learning computer science, but it does not explain why they never take an interest.

Exposure to computers is another component to be considered when looking at the gender gap in computer science. When professionals who p were asked about their exposure to computer science, both males and females did not differ on the amount of exposure (Ehrlinger et al., 2018). Due to this, exposure to computers cannot explain the gender gap in computer science. Yet, more males are represented in these careers, and females are considered a minority (Ehrlinger et al., 2018). In one study, 400,000 freshmen in the US were surveyed about computer use, and results showed there was no difference in computer use, but there was a big difference in confidence between males and females when using a computer (Rubio et al., 2014). Additionally, females entering college intending to major in computer science has decreased from 37% (1980’s) to 14% (2013), so exposure cannot explain why females are not pursuing careers in computer science (Rubio et al., 2014).

A third component to consider why there are fewer females in computer science has to do with perceptions and identity (Rubio et al., 2014). It is believed that these components play a more significant role in who takes an interest in computer coding (Rubio et al., 2014). Television shows, movies, and firsthand experiences, such as exposure in the classroom, play a role in developing stereotypes of a computer scientist (Ehrlinger et al., 2018). The stereotype of a computer scientist is often described as male, intelligent, unattractive, lacking social skills, technology-oriented, and obsessed with computers (Starr, 2018).

It is believed that identity can explain why more women are not pursuing careers in computer science. This is believed to be due to lack of confidence, lack of parental encouragement, and unrelatable stereotypes (Ehrlinger et al., 2018). Stereotypes reduce women's identification with STEM fields, such as computer science, decreasing their motivation to enter those domains. “Stereotypes may be gender-based (STEM is for men) or trait-based (STEM is for geniuses)” (Starr, 2018, p. 489). As a result, females may be more prone to enter specific STEM fields, such as biology, but not others that are more prone to nerd-genius stereotypes such as computer science (Starr, 2018).

Furthermore, compared to males, females have a more significant stereotype of a computer scientist (Ehrlinger et al., 2018). In a study by Ehrlinger et al. (2018), researchers compared male versus female stereotypical views in computer science and engineering fields. The study used two hundred sixty-nine college students, one hundred eighty-seven of whom were females (Ehrlinger et al., 2018). First, the researcher asked the participants to describe the prototypical computer scientist (study 1) and engineer (study 2) through open-ended descriptions as well as through a set of trait ratings (Ehrlinger et al., 2018). Afterward, participants rated themselves, based on the same set of traits, on a scale of one to nine (Ehrlinger et al., 2018). Finally, Ehrlinger et al. (2018) asked the participants to explain their likelihood of pursuing future college courses and careers in computer science (study 1) and engineering (study 2). Results revealed that women held a more stereotypical view of both computer scientists (study 1) and engineers (study 2) than males (Ehrlinger et al., 2018). Additionally, females did not relate to these fields as much as males, based on the set of traits they rated to themselves (Ehrlinger et al., 2018). Finally, Ehrlinger et al. (2018) discovered females were less likely to pursue careers in computer science (study 1) and engineering (study 2) compared to males. This shows females' identities, and stereotypes impacted their motivation to pursue a career in STEM fields like computer science (Starr, 2018).

Due to the negative impacts identity has on females and computer science, how computer science is being introduced and taught needs to change (Rubio et al., 2014). Interventions can be used when designing lessons and surveys (Cromley, Perez, Wills, Tanaka, Horvat, & Agbenyega, 2013). First, when designing introductory computer coding courses, the designers should consider the different perceptions/identities/stereotypes of those who study computer science (Rubio et al., 2014). Harvey Mudd College has a three-pronged approach that has improved female participation in computer science: separate tracks based on experience, computing research experiences, and female community-building activities (Rubio et al., 2014).

Additionally, Ehrlinger et al. (2018) explained a way to overcome stereotypes of a computer scientist is to give young men and women direct experiences with counter-stereotypic exemplars in the field. By doing this, females are being exposed to someone who does not match their stereotype and is in the field of computer science. Another way to reduce nerd-genius stereotypes for women in computer science is to remove artifacts reminiscent of these stereotypes, such as a picture of a male computer scientist (Starr, 2018). Additionally, Toglia (2013) has five ways to improve female participation in computer science: (1) mentoring programs, (2) removing gender depicting content materials, (3) implementing parent education programs, (4) having counselors receive ongoing training to ensure they have the tools to promote STEM careers for females (5) bringing female guest speakers of STEM careers into the classroom (Toglia, 2013). Finally, Google launched "Made with Code," which targeted females and focused on coding projects, female community-building activities, and video profiles of women in computing (Rubio et al., 2014).

2.1.2 Theme Two: Essential Thinking Skills

The second theme that emerged when reviewing past literature was thinking skills needed for students to become proficient in computer coding. Two main thinking skills were frequently discussed, including Computational Thinking and Algorithmic Thinking. Both thinking skills were often mentioned as needed skills for those not only interested in pursuing computer coding, but as general education skills (Futschek, 2006; Lee & Junoh, 2019; Sung, 2019).

Computational Thinking is a cognitive process where an individual takes a large problem and divides the problem into smaller sections that can be solved through a set of steps or a flowchart (Lee & Junoh, 2019; Ricketts, 2018; Sung, 2019; Tonbuloglu & Tonbuloglu, 2019). Computational Thinking has become popular recently, particularly in computer coding education. Sung (2019) states that Computational Thinking has a close relationship with technology and engineering education because similar processes occur. "The use of technology involves Computational Thinking skills, and computer science is used for the acquisitions of Computational Thinking skills" (Tonbuloglu & Tonbuloglu, 2019, p. 404). In Computational Thinking, problems are broken down into smaller sections through a flow chart. In computer science, the same thing occurs, but it is called decomposition and illustrates the logic from the start to the end when solving a problem (Sung, 2019). Critical elements of Computational Thinking include (1) abstraction and automation (2) systematic process and information (3) symbol systems and representations (4) algorithmic notions of flow control (5) structured problem decomposition (6) iterative, recursive, and parallel thinking (7) conditional logic (8) Efficiency and performance constraints (9) Debugging and systematic error detection (Grover & Pea, 2013). In Table 2.1, an example of Computational Thinking can be seen. The question of how many golf balls could a boat model hold is the guiding question, and the table shows the steps, or computation thinking, involved in finding the answer (Sung, 2019). The problem is broken down or decomposed to make a flow chart of the solution that shows the logic and thinking behind solving the problem.

Table 2.1

Example of Computational Thinking Practices (Sung, 2019, p. 12)

Guiding Questions	Use of Computational Thinking
1) What is the weight of a golf ball?	Weight of a golf ball using a scale. • *m^golf ball = 46g*
2) What is the total weight of 20 golf balls?	Build a formula to get the total weight. • *m^golf ball x 20 = 46g x 20 = 920g*
3) What is the formula to get the volume of a boat that holds 920g?	Complete the volume formula that meets the critical load using the density triangle. • *Vcritical load = 920g/Dwater* • *Dwater = 1g/1ml* • *Vcritical load = 920cm3, 1ml = 1cm*
4) What is the minimum length, depth, and height for the critical load?	Compute the volume of your design model. • *920cm3 < Lcm x Dcm x Hcm*
5) How would the critical load be illustrated in a graph using the formula? The graph should include two lines representing critical density numbers and minimum loads.	Illustrate a graph that represents the critical density of water and minimum loads
6) What is the theoretical number of golf balls that your boat model loads?	• *mmax load = Dwater x Vraft model* • *e.g.) L = 15cm, D = 15cm, H = 5cm, VLXDXH = 1125cm3* • *mmax load = 1125g* • *Predicted number of golf balls = 1125g = 24.46*

Algorithmic Thinking is also often associated with computer coding and is a set of abilities used to construct and understand algorithms that will complete a task or solve a given problem (Lee & Junoh, 2019). Using algorithms is an underlying theme in engineering, computer science, and other “hard science” such as physics, chemistry, biology, and astronomy (Hurlburt, 2018). A simple definition of an algorithm is a list of step-by-step procedures used to complete a task (Lee & Junoh, 2019). Algorithms are often associated with computer coding (Lee & Junoh, 2019). Learning to computer code involves Algorithmic Thinking because computer coders need to know to develop a set of instructions, or an algorithm, to complete a task or solve a given problem (Ricketts, 2018). Algorithmic Thinking includes decomposition, repetition, data organization, generalization, parameterization, algorithm vs. program, top-down design, refinement, and critical thinking (Cooper et al., 2000; Hurlburt, 2018). Critical thinking is needed when applying mathematical knowledge to create an algorithm to solve a problem. This skill goes beyond the classroom and into many aspects of life (Hurlburt, 2018). By improving critical thinking and Algorithmic Thinking, students learn what makes up an algorithm, how to appreciate them, and eventually how to develop new algorithms (Hurlburt, 2018).

Unfortunately, by the time students enter college, they can often not construct a stepwise algorithm to solve a given problem (Cooper, Dann, & Pausch, 2000). As a result, if a teacher wants to instill Algorithmic Thinking into students, the process should be started at a young age when the students are children (Cooper et al., 2000). An example of this is identifying daily tasks that implement an algorithm design, such as washing hands, brushing their teeth, or putting on a jacket (Lee & Junoh, 2019). By looking at these tasks, students can explain how each job is carried out, which is an example of Algorithmic Thinking.

Computational and Algorithmic Thinking are sometimes viewed as interchangeable ideas that are not defined clearly but are related. Algorithmic Thinking can be thought of as a part of Computational Thinking. A weakness of Algorithmic Thinking is that mathematical formulas only provide the minimum requirements for solving a problem and do not consider the logic and reasoning behind the steps that Computational Thinking does (Sung, 2019). In other words, Algorithmic Thinking creates an algorithm, or a series of steps, to solve a given problem. Computational Thinking is more about developing skills on how to approach a problem. One of the computational skills is developing an algorithm through Algorithmic Thinking. Computational Thinking helps understand how to approach all problems, while Algorithmic Thinking is the development of an algorithm for a particular problem.

Learning a computer coding language can be intimidating for a beginning computer coder and may cause a lack of interest due to this difficulty. As a result, the computer coding language barrier needs to be removed and replaced with a variety of problem-solving experiences that require the use of decomposition and algorithms (Sung, 2019). Once students have improved their computational and Algorithmic Thinking, then computer coding languages can be incorporated. Learning to approach problems and create an algorithm to solve the problem is an important skill.

According to Lee and Junoh (2019), when incorporating these thinking skills into lessons for kids, they should be based around an underlying story, just as a computer code would be based on an underlying task. Once students are given a task, several steps should occur, including (1) students being given a variety of sizes of blank grid paper and directional arrow cards, (2) students being given appropriate props to act out their story (3) students being given writing and drawing materials to record their stories in a step-by-step format (4) commercialized coding toys being integrated into instruction to provide opportunities to computer code and see the results (5) the teacher incorporating computer code-associated terms when discussing results to build meaning in students (Lee & Junoh, 2019).

2.1.3 Theme Three: STEM vs. STEAM

The third theme that emerged when reviewing research was adding art into STEM. STEM is an acronym for Science, Technology, Engineering, and Math, while STEAM adds a fifth component, Art. By adding art, a design component is being added that may aid STEM fields (Costantino, 2018). By using art in a transdisciplinary approach to teaching STEM, an interest that wasn't in an individual before may develop (Costantino, 2018, p. 102). The goal of adding art to STEM is to add creativity and innovation to the field (Daugherty, 2013). Research shows the art component must be meaningful and not just for entertainment purposes (Adkins et al., 2017). Additionally, the idea of STEAM has shown evidence of improving cognitive scores, increasing innovation, and motivating students into STEM (Colucci-Gray et al., 2017).

Additionally, Peppler and Glosson (2013), a leader in incorporating art into STEM, believes the STEAM approach promotes more female and minority involvement in technology-related disciplines (such as computer education). A "STEAM-powered approach to education aims to balance technical expertise with artistic vision" (Peppler & Glosson, 2013, p. 39).

There have been several studies that have been carried out using the STEAM approach (Adkins et al., 2017; Gulhan & Sahin, 2018; Jawad et al., 2018; Jawad et al., 2018; Looi et al., 2018; Thuneberg et al., 2017). These studies have a variety of designs but focus on incorporating art into a STEM concept. The studies range from elementary to graduate level and examine the impacts of adding art on understanding, motivation, and attitude towards STEM. There is no correct design, and the experiments were built based on what the researchers wanted to study.

In a study by Gulhan and Sahin (2018), the value of adding art as a design aspect into STEM was examined. Thirty students in the seventh grade participated in five STEAM activities over the reflection and absorption of light. The lesson took five weeks and a total of twenty hours (Gulhan & Sahin, 2018). Lessons ranged from experimenting with different types of mirrors, designing kaleidoscopes, and testing with reflecting a rainbow. After the activity, Gulhan and Sahin (2018) examined projects that combined elements of the engineering design process, art, and science. They noticed positive results that combined a creative design linked with the science content. Six students were also interviewed after the activity. Gulhan and Sahin (2018) asked students if adding arts increased their interest in STEM fields, and students responded that, yes, art helped them understand science concepts better, and the communication and teamwork in the design process were helpful. Finally, when Gulhan and Sahin (2018) asked which fields STEAM students enjoyed the most, four of the six students replied that art was their favorite field. A new group of students were brought into STEM by combining art into STEM.

In another study by Adkins et al. (2017), the benefits of the interdisciplinary fusions of science and art were examined through a case study that lasted one semester in a college setting. In this study, agar, a jelly derived from algae used to grow bacteria, art was used in an introductory microbiology course, and results were compared to a control course using the traditional format of teaching microbiology (Adkins et al., 2017). The control group consisted of thirty-three participants, and the experimental group consisted of thirty participants. Both groups of students received soil samples at the beginning of the semester and had to isolate and identify different microbes from the soil samples. The courses were compared by students' ability to identify an unknown microbe, pre-and post-surveys, and blind interviews to measure concept mastery, attitudes towards science, and student demographic characteristics. Adkins et al. (2017) discovered that students who participated in open-inquiry agar art activity had greater confidence in their ability than the control class. Although, both classes performed at the same level. Even with the small sample size, the results suggest that incorporating art in an intentional format can enhance a STEM course, students' understanding, confidence, and desire to continue studying the topic (Adkins et al., 2017).

In a study by Peppler and Glosson (2013), the use of an e-textile toolkit and if it could aid youth in learning about electronics in an elective setting and whether e-textiles could elucidate important circuitry concepts that traditional materials have historically struggled to convey was studied. The research had two goals: to determine how youth learn about electrical circuits in an elective environment and how can e-textiles facilitate the learning of important concepts in electrical circuits that traditional materials have historically struggled to convey (Peppler & Glosson, 2013). Overall, Peppler and Glosson (2013) tested whether youth could create an overall working circuit by whether they understood three core concepts: current flow, connections, and polarity by combining art and science. The study by Peppler and Glosson (2013) focused on seventeen youth between the ages of seven and twelve at a local Boys and Girls Club. The twenty-hour after-school program that met two times a week for two hours each was designed to teach the youth about electrical circuitry and then create an e-textile project combining the art of the e-textile with the science of circuitry. Peppler and Glosson (2013) collected data through pre-and post-tests to gauge students' understanding of current flow, connection, and polarity. Afterward, data was analyzed using a sequential mixed-methods approach (pre-post with paired samples, videotaped observations, artifacts, circuit design assessment). The circuit design assessment was analyzed by coding the youths' responses on a five-point scale (Peppler & Glosson, 2013). Peppler and Glosson’s (2013) results revealed that the combination of art and science did produce positive results, including that youth were engaged in the e-textile design, demonstrated gains in their ability to produce a model drawing of a working circuit, and could explain how current flow, polarity, and connections worked.

In a fourth study by Jawad et al. (2018), integrating art and animation in teaching computer coding to high school students was explored. The program combined art, animation, and computer code using three groups of high school students. Each group received instruction in a different setting, and hours of instruction ranged from three to twenty-five hours. The study investigated students’ interest, knowledge of computer coding, and their interest in pursuing a degree in computer science (Jawad et al., 2018). Data was gathered through pre-and post-test surveys using a five-point Likert scale to measure students' computer science degree interest (Jawad et al., 2018). Jawad et al. (2018) determined through the study that by combining art with science, students’ knowledge, enjoyment, motivation in learning computer coding, and their interest in pursuing a degree in computer science after graduation increased. Surprisingly, there was a more significant increase in females than males, and this new approach combining art and science was hypothesized why (Jawad et al., 2018). Most participants were excited to share their art at the completion of the course. In an additional case study by Hunter-Doniger and Sydow (2016), the effects of changing a STEM curriculum to a STEAM curriculum in middle school was examined. The purpose of the study was to investigate the effectiveness, importance, and sustainability of a STEAM curriculum. The STEAM curriculum should be interdisciplinary and inquiry-based to promote creativity, problem-solving, critical thinking, and collaboration (Hunter-Doniger & Sydow, 2016). The school the study occurred in had seven hundred and seventy-six students in grades sixth through eighth. Data was collected through Likert scale surveys given to teachers at the beginning and end of the study, field notes and observations from teachers, interviews from teachers, and demographic surveys. Data was analyzed through mixed methods and driven by two research questions. Results showed that adding art to STEM was valuable in staff satisfaction and student learning. Hunter-Doniger and Sydow (2016) plan to continue their study for two additional years and focus more on students' experience using surveys, focus groups, and test scores.

Finally, in a study by Thuneberg et al. (2017), art was combined with math to create an exhibition. The study used two hundred and fifty-six students between the ages of twelve and thirteen. Students completed the validated SQR-A and RAVEN questionnaires pre-and post-the study (Thuneberg et al., 2017). Data was analyzed through linear modeling and structural equation path modeling. Based on these results, students were grouped into performance groups. Results were positive, with even the lowest-scoring group appreciating the math and art concept (Thuneberg et al., 2017). Overall, the attitude was positive in both the experience and the content learned.

2.1.4 Theme Four: Pedagogical Teaching Approaches

Today’s students were born in a digital-dependent era, and they are used to technology being a guiding component in their lives (Gurbuz et al., 2016). Additionally,

> Children frequently experience automated systems with coding-based systems in their daily lives. Due to rapid changes in technology, children are being exposed to these systems more and more, and this exposure naturally promotes their interest in how things perform or move automatically (Lee & Junoh, 2019, p. 709).

Coding is a language, and today’s students need to become literate in this language (Lee & Junoh, 2019). Digital literacy involves not just the ability to chat, browse, and interact online but also the ability to design, create, and invent with new media (Resnick et al., 2009). Much of the population view computer coding as only appropriate for a specialized few, but that is not the case anymore (Resnick et al., 2009). Initially, computer coding language programs were too difficult for the beginning computer coder because they involved challenging activities with no context for the learner (Resnick et al., 2009). Now, different pedagogical approaches include computer coding “unplugged” and a variety of coding instructional tools.

Computer coding “unplugged” has existed for over twenty years. Bell, Witten, and Fellows (1998) conducted some of the earliest research on coding “unplugged” and created several “unplugged” coding activities. Bell et al. (1998) combined those activities in a free online book called, “Computer Science “unplugged”: Off-line Activities and Games for All Ages”, which presented the idea that computer coding could be taught without a computer (Bell & Vahrenhold, 2018). In all formats, the computer coding “unplugged” approach has promoted the same core principles, including: (1) the barrier of learning a computer coding language is removed (2) activities are meaningful and applicable (3) learning occurs without the use of a computer (4) content can be taught in a variety of settings with a fluctuating number of participants (Bell & Vahrenhold, 2018).

These activities have recently been published on a website called CSunplugged.org (Bell & Vahrenhold, 2018). These activities have now been translated into several languages and are used in numerous curricula such as code.org, Hour of Code, Discover Project – I Discover Coding, “unplugged” Coding Game, and Beaver (Tonbuloglu & Tonbuloglu, 2019). Additionally, coding “unplugged” has continued to evolve as more teachers/developers have created new activities to contribute to this approach of teaching computer coding (Bell & Vahrenhold, 2018). As the evolution has occurred, the focus has also expanded, and now coding “unplugged” has become known as more of a pedagogical approach for introducing computer coding and computer science (Bell & Vahrenhold, 2018). The appeal of “unplugged” computer coding is that the activities provide a format to teach computer coding that allows for collaboration, creativity, and problem solving (Cortina, 2015). Using “unplugged” techniques, complex ideas can be taught in an engaging, fun atmosphere that uses minimum time (Bell & Vahrenhold, 2018).

Coding “unplugged” was created for outreach programs, such as camps or after school events so that students could learn to code even when computers were not in supply (Bell & Vahrenhold, 2018). Due to this approach, coding “unplugged” could reach a broader and larger audience in a variety of settings (Bell & Vahrenhold, 2018; Tonbuloglu & Tonbuloglu, 2019). Since computers are not needed, there are no restrictions on the number of participants based on the availability of materials. Additionally, “unplugged” can teach more than computer coding, including Computational Thinking, collaboration, and problem-solving (Tonbuloglu & Tonbuloglu, 2019).

There are several requirements for an activity to be considered “unplugged”. These requirements include that no computers can be used, the ideas being taught are based on real computer science/coding, participants learn by doing, the activities are fun, the activities may be based on a game to promote engagement, there is a kinesthetic component, and the activities require resilience, including trial and error, to solve (Bell & Vahrenhold, 2018; Tonbuloglu & Tonbuloglu, 2019). Additionally, “unplugged” activities usually use inexpensive and widely available materials that would be easily accessible (Tonbuloglu & Tonbuloglu, 2019).

There are several ways that “unplugged” coding can be approached. Due to the flexibility of this teaching method, this approach can be taught in both traditional (classroom) and non-traditional (camps, after-school programs, etc.) settings (Bell & Vahrenhold, 2018). An example of how to introduce the concept of coding in an “unplugged” format can be as simple as asking about daily tasks. First, the instructor needs to ask students/participants what tasks they complete daily, such as brushing their teeth, washing their hands, etc. (Lee & Junoh, 2019). This approach is used because computer coding creates a list of steps to complete a task, similar to a list of steps to complete a daily task (Lee & Junoh, 2019). In either case, students create an algorithm comprised of code, or a list of steps, to complete a task. Computer coding may be a new term to the students/participants, so students may be able to relate by approaching the concept of computer coding with familiar concepts, such as washing hands (Lee & Junoh, 2019). Students can also debate if there are different or better ways to complete a task, and from there, a discussion can be held on debugging, removing unnecessary steps, loops, and errors (Lee & Junoh, 2019). Students can also practice coding daily, outside of the instructional setting, by thinking about the steps involved to complete a given task (Lee & Junoh, 2019).

Once students have grasped the idea that their daily tasks involve a set of steps, students/participants can be challenged to draw these steps physically on pieces of paper (Lee & Junoh, 2019). Once completed, students can then take the drawings and use directional arrows lay out their steps/code on a piece of paper (Lee & Junoh, 2019). The pictures serve as a manipulative that can be moved and re-arranged to design the best algorithm (Lee & Junoh, 2019). Students/participants can also switch their algorithms with other students and challenge each other to figure out what task their algorithm is showing. There are other examples of “unplugged” coding, such as providing a piece of grid paper with a start and end and having students draw directional arrows in each block to complete the tasks (Lee & Junoh, 2019). Additionally, students can direct each other using directional words to get one participant from one location to another. Teachers/instructors can also read simple stories and have students draw/describe the steps involved to complete the algorithm of the story (Lee & Junoh, 2019). All these examples involve the combination of real-life application, a non-intimidating approach, and collaboration to teach coding.

There are a couple of defined limitations of “unplugged” coding. First, since coding “unplugged” was initially designed as an outreach tool to introduce students to computer coding, “unplugged” coding is not meant to be taught by itself beyond the outreach (Bell & Vahrenhold, 2018). In other words, “unplugged” coding is not a curriculum or program of study, but it is instead a supplement/introductory tool to a computer coding curriculum (Bell & Vahrenhold, 2018). Additionally, as coding “unplugged” has become more common, the pedagogical definition has become contradictory if plugged coding must be involved when teaching “unplugged” computer science (Bell & Vahrenhold, 2018). Third, coding “unplugged” can only teach general ideas of computer science but not details (Bell & Vahrenhold, 2018). Coding “unplugged” should be based on general concepts, which is why coding “unplugged” is considered a supplement. Another weakness of “unplugged” coding is that it starts from scratch when it is taught and is not linked to any previous student experience (Bell & Vahrenhold, 2018). With some students, this may become a problem, and they may lose interest due to their belief that they are already experts (Bell & Vahrenhold, 2018). With these limitations, “unplugged” coding is still thought of as an effective practical approach to coding (Bell & Vahrenhold, 2018).

When defining “effective” concerning “unplugged” coding, two main categories are often referred to in research: an improvement in attitude towards coding and the advancement of knowledge of coding (Bell & Vahrenhold, 2018; Lee & Junoh, 2019;

Tonbuloglu & Tonbuloglu, 2019). Research has revealed both successes of “unplugged” coding and instances when it may not have been the best pedagogical approach. One thing to consider when reading is that the research conducted rarely examines the effectiveness of the overall pedagogical approach to “unplugged” coding, but it is instead focusing on the effectiveness of a specific program under the umbrella term of coding “unplugged” (Bell & Vahrenhold, 2018).

The first defining element of the effectiveness of coding “unplugged” is an improvement in attitude towards coding and computer science (Bell & Vahrenhold, 2018). Since “unplugged” coding originated as an outreach program, many view the improvement of attitude towards coding and computer science as the primary goal of the “unplugged” approach (Bell & Vahrenhold, 2018; Lee & Junoh, 2019; Tonbuloglu & Tonbuloglu, 2019). Taub et al. (2012) define the attitudes of coding as “representing evaluations towards an “attitude object” in dimensions such as good/bad, harmful/beneficial, pleasant/unpleasant and likable/unlikable; for example, evaluating computer science as boring or tedious” (p. 8:5). Additionally, Taub et al. (2012) define attitude to “include the motivational factors that influence a behavior” (p. 8:5). This includes the motivation to pursue a study of computer science. Results from research are mixed on if “unplugged” coding improves attitude.

In a study by Taub et al. (2012), the attitudes, understanding, and achievements of “unplugged” coding were studied in two traditional and one non-traditional classroom settings using the *CSunplugged* book by Bell et al. (1998). Seventh-grade students (ages 12-13) were used because the researchers felt that it was the optimal age when students started making decisions about their future studies. The study was mixed methods and included a Likert-type questionnaire as the quantitative data and open-ended question interviews as the qualitative data (Taub et al., 2012). Data was collected both before students were exposed to the “unplugged” activities and after the program, after students were exposed to the “unplugged” activities. Two schools and teachers with no prior computer coding education participated in the study. The first school, an all-girls school, studied the “unplugged” activities over a semester using two different classes (N1 = 27, N2 = 25). The second school, a co-ed school, studied the “unplugged” activities in an obligatory after-school program (N3 = 26). All three classes spent two hours a week studying the same “unplugged” activities. Taub et al. (2012) instructed the teachers to use the activities that covered binary numbers, image representation, text compression, error detection, information theory, searching algorithms and sorting algorithms (activities one to seven). Additionally, in the first school, the students also studied the activity that dealt with graph coloring (activity fourteen), while in the second school, they studied sorting networks (activity eight).

Data was analyzed using several methods. First, questionnaire data was analyzed using a *t*-test from before the study and after the study. Then a one-way ANOVA test was used on the results from the questionnaire to determine differences in the scores from each class (Taub et al., 2012, p. 8:8). Data from interviews about views was coded into different categories, including “(a) the nature of computer science, (b) women in computer science, (c) the work in computer science (specifically, cooperation in computer science), (d) careers in computer science, and (e) the relation between computer science “unplugged” and concepts in computer science” (Taub et al., 2012, p. 8:8). Data from interviews about attitudes was data coded into different categories including "(a) attitudes toward computer science and their perceived future success, (b) attitudes toward the computer science "unplugged" activities, and (c) attitudes towards the computer scientist" (Taub et al., 2012, p. 8:8).

Results from this study had the opposite effect the researchers were expecting. After the study, Taub et al. (2012) found that students' attitudes towards computer coding and computer science had decreased. Participants found the topic of computer coding less interesting, and they had less of a desire or motivation to pursue a career in computer science. This was the opposite result Taub et al. (2012) was expecting. There were some restraints the researchers identified that may have caused these unexpected results. First, only a couple of activities were covered due to time constraints (Taub et al., 2012). If more time had been allotted, students would have completed more activities and become more engaged. Second, Taub et al. (2012) believe that outside interactions of students impacted views, attitudes, and intentions. Finally, the researchers thought that the students didn't connect the purpose of activities with coding (Taub et al., 2012). Even with these unexpected results, Taub et al. (2012) still believe that "unplugged" coding has potential and should be used, and further research is needed.

In another study, Tonbuloglu and Tonbuloglu (2019) conducted a study in a traditional classroom setting on how coding activities impacted the Computational Thinking of middle school students. The guiding questions of the study asked if "unplugged" activities impacted Computational Thinking skills and what were the experiences of students when learning "unplugged" activities (Tonbuloglu & Tonbuloglu, 2019). The mixed-method study used pre- and post-surveys before and after students experienced ten weeks of “unplugged” learning along with observations and daybooks/student journals. One hundred and fourteen students in fifth grade participated. The survey used was the “Computational Thinking Skill Levels Scale” developed by Korkmaz, Cakir, and Ozden (2017), and utilizes a five-point Likert type scale over twenty-two Computational Thinking Skills with the lowest score being twenty-two and the highest score being one-hundred and ten (Tonbuloglu & Tonbuloglu, 2019). The observations and daybooks allowed the researchers to monitor the class daily. Observations and daybooks included field notes of experiences, activities performed, reactions, the flow of the lessons, areas of difficulty, and notes about the result of the lesson (Tonbuloglu & Tonbuloglu, 2019). The survey was analyzed through a paired t-test and observations and daybooks were analyzed through content analysis (Tonbuloglu & Tonbuloglu, 2019).

Results from this study revealed different results from the Taub et al. (2012) study. The student’s reactions were determined through the observations and daybooks/student journals and revealed the students found the lessons entertaining and motivating (Tonbuloglu & Tonbuloglu, 2019). As a result, Tonbuloglu and Tonbuloglu (2019) believe that “unplugged” activities have a positive effect on attitude and the motivation to pursue a career in computer science. Tonbuloglu and Tonbuloglu (2019) also experienced limitations of their study including the overcrowding of class size which caused scheduling conflicts with the activities.

Another theme that has emerged around the effectiveness of “unplugged” coding is the improvement of knowledge about computer coding/computer science concepts. The main mention of knowledge skills that are studied, regarding “unplugged” coding, are Computational Thinking skills. Computational Thinking is a cognitive process where an individual takes a big problem and divides the problem into smaller sections that can be solved through a set of steps or a flowchart (Lee & Junoh, 2019; Ricketts, 2018; Sung, 2019; Tonbuloglu & Tonbuloglu, 2019). Computational Thinking has become popular recently, particularly in coding education. Sung (2019), states that Computational Thinking has a close relationship with technology and engineering education because similar processes occur. "The use of technology involves Computational Thinking skills, and computer science is used for the acquisitions of Computational Thinking skills” (Tonbuloglu & Tonbuloglu, 2019, p. 404). In Computational Thinking, problems are broken down into smaller sections through the process of a flow chart. In computer science, the same thing occurs, but it is called decomposition and illustrates the logic from the start to the end when solving a problem (Sung, 2019). Key elements of Computational Thinking include (1) abstraction and automation (2) systematic process and information (3) symbol systems and representations (4) algorithmic notions of flow control (5) structured problem decomposition (6) iterative, recursive, and parallel thinking (7) conditional logic (8) Efficiency and performance constraints (9) Debugging and systematic error detection (Grover & Pea, 2013).

Looi, How, Longkai, Seow, and Liu (2018) conducted a mixed-methods two-phase design study in a traditional educational setting to understand how “unplugged” activities impact the development of Computational Thinking. This was done by looking at the "in-between" process from using the “unplugged” activities, creating an artifact, and evaluating the types of Computational Thinking skills developed (Looi et al., 2018). Two groups of students (activity group and control group) underwent an introduction to computing class, one group completed the program in the traditional format (n = 18) and one group completed the program in the “unplugged” format (n=17). In the end, both groups were asked to produce an artifact to represent their knowledge of a sorting algorithm (Looi et al., 2018). Additional data was collected through interviews and video recordings of groups during the lessons. The data was coded, and inter-rater reliability tests were conducted.

Results revealed that the Computational Thinking scores were higher for the students that participated in the “unplugged” activity group compared to the traditional formatted class (Looi et al., 2018). Additionally, the individual computation thinking skills (decomposition, algorithmic design, generalization, abstraction, and evaluation) were broken down revealing that twelve of the seventeen students participating in the ““unplugged”: curriculum used all five of the Computational Thinking skills compared to the control group where only three of the eighteen used all five of the Computational Thinking skills (Looi et al., 2018). The limitations of this study included a small sample size, and the analysis of data did not consider how the interactions of factors may impact the outcome (Looi et al., 2018).

Additionally, there has been more data to support the idea the “unplugged” coding does improve Computational Thinking. In the Tonbuloglu and Tonbuloglu (2019) study, besides “unplugged” activities increasing attitude, results between the pre- and post-survey also showed improvement of Computational Thinking, creativity, Algorithmic Thinking, collaboration, and critical thinking. Also, Taub et al. (2012), while stating the "unplugged" coding hurt attitude, revealed an increase in skills needed for Computational Thinking from the beginning of the study to the end. None of these studies suggested that using "unplugged" coding had no effect or hurt students' understanding of Computational Thinking.

The main noticeable difference between plugged coding versus "unplugged" coding is that plugged coding uses a computer (Lee & Junoh, 2019). As mentioned earlier, coding is, "the process of identifying and labeling each step to complete a task" (Lee & Junoh, 2019, p. 712). The traditional method of teaching coding is by creating an algorithm using codes on a computer. There are multiple computer languages such as Python and C++ that can be used to create the code (Papadakis et al., 2016; Resnick et al., 2009). For this method, the focus is on completing the given task, compared to "unplugged" coding, where the focus is on building the skills needed to understand how to complete the task (Lee & Junoh, 2019; Tonbuloglu & Tonbuloglu, 2019). Plugged coding is often referred to as just computer coding, but for differentiation in this paper, it will be referred to as plugged coding (Lee & Junoh, 2019).

As mentioned earlier, the reason that original plugged coding instruction lost interest was due to coding languages being difficult for students to understand and master, coding being introduced using activities that had no meaning in a child's life, and the restrictions of coding did not allow for guidance when the code did not work (Resnick et al., 2009). While one method to overcome these challenges was the creation of “unplugged” coding, another method was to change how coding instruction was presented (Lee & Junoh, 2019; Rubio et al., 2014).

There are a variety of coding tools, environments, and approaches that aid the beginning coder in learning (Gurbuz et al., 2016). First, there has been a fluctuation in resources and curriculum that focus on teaching coding such as hour-of-code, code academy, scratch, app inventor, Alice, and light-bot (Rubio et al., 2014). Another emerging coding resource is using a toy or robot that can be coded to carry out certain tasks. Lee and Junoh (2019) state that, “integrating commercialized coding toys provide opportunities to code and to see the coding toys or robots move based on the algorithm they created and input” (p. 715). Contextualization is also important. Instead of writing an abstract program, students can learn about basic programs through writing codes that have meaning to them such as coding a robot to exit a maze, animating a story, playing a game, or creating light symphonies (Rubio et al., 2014).

A leading approach to teaching coding, especially to younger students, is through drag-and-drop programs (Resnick et al., 2009). As mentioned before, there should be a focus on teaching young students coding because they develop an interest at a young age, and they need to develop the skills to pursue coding at a young age (Lee & Junoh, 2019). Drag-and-drop programs use virtual colorful boxes that look like long puzzle pieces and represent different commands (Resnick et al., 2009). Drag-and-drop programs are also referred to as block-based coding. This method of coding eliminates the need for students to type code (Papadakis et al., 2016). Block-based or drag-and-drop coding uses a menu that contains blocks that are categorized and color-coded. Students have a choice of these blocks, or codes, to complete the task. Once a block is chosen it is fitted with another block. If the code is possible the blocks will fit together like a puzzle piece, but if not, they will not link (Papadakis et al., 2016). By fitting together different commands different actions occur (Resnick et al., 2009). Block-based/drag-and-drop coding removes the barrier or learning a coding language and focuses on improving conceptual thinking (Papadakis et al., 2016). Resnick et al. (2009) believes drag and drop would interest children due to media creation, scaffolded approach, and colorful visuals. Due to the range of options for teaching plugged coding, drag-and-drop/block-based programs will be the main focus when discussing plugged coding in this paper.

Logo, the first block-based language, was developed in 1995, but Scratch, developed in 2007 at MIT, is more well-known and popular (Papadakis et al., 2016). Scratch is a computer coding language that uses the block-based/drag-and-drop approach. The Scratch website (http://scratch.mit.edu) launched in 2007 and consists of a community of coders that create projects using the Scratch version of block-based/drag-and-drop approach (Resnick et al., 2009). Scratch has been called "the YouTube of interactive media" (Resnick et al., 2009, p. 1). Anyone can gain access to the Scratch website, free of charge, and write code to create "video games, interactive newsletters, science simulations, virtual tours, birthday cards, animated dance contests, and interactive tutorials" (Resnick, et al., 2009, p 1). While Scratch is open to participants of any age, the focus is on individuals between the ages of eight and sixteen, with the core audience being twelve (Resnick et al., 2009).

Papert (1980, as cited in Resnick et al., 2009) states that computer coding languages should have a “low floor” (easy to get started) and a “high ceiling” (opportunities to create increasingly complex projects over time). In addition, languages need “wide walls” (supporting many different types of projects so people with many different interests and learning styles can all become engaged). “Satisfying the triplet of low-floor/high-ceiling/wide-walls hasn’t been easy” (p. 4).

Resnick et al. (2009) believes that while some other computer coding languages have tried, such as Flash and Alice, Scratch can meet these needs the best for making computer coding accessible for all. Resnick et al. (2009) believe that Scratch is “more tinkerable, more meaningful, and more social than other computer coding environments” (p. 4). One Scratch user that has achieved some fame is BalaBethany. BalaBethany already enjoyed drawing anime characters on paper and made the transition to animating them on Scratch (Resnick et al., 2009). She would post her designs on Scratch and get glowing reviews along with questions on how she achieved her computer code to make her drawing. This prompted her to make episodes using her characters, and she even created competitions where other users would send it designs for her to use. Additionally, some users stated they didn’t know how to computer code an anime character so BalaBethany computer coded videos on how to draw her characters. In a year BalaBethany computer coded over 200 projects on Scratch (Resnick et al., 2009). This is an example of how Scratch has enabled a user to take her interests and expand upon them using Scratch.

While Scratch/drag-and-drop/block-based programs seem promising there are some limitations. First, the goal of these programs is not to prepare students for a career is computer coding (Resnick et al., 2009). Like "unplugged" coding, Scratch/drag-and-drop/block-based programs desire to introduce young students to coding in a nurturing non-frustrating format with the hopes that interest will be developed to pursue a career in coding (Papadakis et al., 2016; Resnick et al., 2009). Second, for Scratch/drag-and-drop/block-based programs to be effective students need to be able to transfer the computational skills they developed to current text-based computer coding languages that are used in the professional computer coding world (Weintrop & Wilensky, 2019). Third, when an individual starts computer coding in a text-based format there are no pre-made commands in boxes that a person can choose from. Scratch/drag-and-drop/block-based programs may give false impressions of what is involved in computer coding (Weintrop & Wilensky, 2019). Fourth, Scratch does not use procedures and therefore cannot model recursions, which is an important theme is computer science (Papadakis et al., 2016). Finally, Weintrop and Wilensky (2019) believe that using Scratch/drag-and-drop/block-based programs have no gain in attitude or ability to computer code, so the purpose of using these programs is similar to "unplugged" coding, as an outreach tool for a wide audience.

Even though Scratch/drag-and-drop/block-based are popular, their effectiveness is debatable (Weintrop & Wilensky, 2019). Scratch has made updates since its release in 2007, including allowing users to create their own blocks, store data, export projects, create projects that react in the physical world using a webcam, and sharing multiple levels of granularity (Papadakis et al., 2016). To be able to compare plugged coding to “unplugged” coding the same definition of effective will be used. When defining "effective" concerning plugged coding the two main categories that will be discussed are an improvement in attitude towards coding and the improvement of knowledge of coding. Attitude is how a participant feels towards the activity (Taub et al., 2012) while knowledge is based on the computational skills developed (Ricketts, 2018; Sung, 2019).

Papadakis et at. (2016) conducted research over a four-month period on the effectiveness of Scratch and another coding Android program called AIA. The study involved three groups of students in a traditional education setting, one control group taught basic computer coding (n=18), one experimental group taught AIA (n=35), and another experimental group taught Scratch (n=34). Data was collected in a pre-test, teacher intervention, and post-test. The goal of the study was to evaluate attitudes towards coding using the computer attitude scale (CAS) and to analyze student knowledge using the questionnaire computer coding knowledge (QPK) assessment (Papadakis et al., 2016). Data was analyzed using SPSS and the standard level of significance. To compare the results of the CAS and QPK a paired samples t-test was used along with separate ANOVA tests (Papadakis et al., 2016).

Results from this study showed that the students that used Scratch and AIA improved their attitude towards computer coding and computer science. Additionally, Papadakis et al. (2016) believe that students' feelings towards coding and motivation to pursue a career in computer science is very important. To build this motivation content needs to apply to students' lives (Papadakis et al., 2016). Limitations of this study included a short time frame and the use of only one region in Greece.

Saez-Lopez et al. (2016) conducted a study to determine how Scratch impacts attitude and Computational Thinking skills and practices in a traditional upper elementary education setting. One hundred and seven students participated between the grades of fifth and sixth in five different schools over two years. Each year consisted of a twenty-hour program. The study used a mixed-methods quasi-experimental approach. Data was collected through the Visual Creative Computing Test (VBCCT) and a questionnaire to analyze learning processes and student's attitudes before and after the program (Saez-Lopez et al., 2016). Cohen’s kappa was used to analyze inter-rater reliability and results were compared using a t-test (Saez-Lopez et al., 2016).

Results from this study revealed that the attitude of participants was positive, using words such as "motivating, fun, and enthusiastic" to describe students' experiences (Saez-Lopez et al., 2016, p. 139). In both studies students’ attitudes were positive (Papadakis et al., 2016; Saez-Lopez et al., 2016). This suggests that using Scratch can improve an individual’s attitude towards coding and computer science. This goes against what Weintrop and Wilensky (2019) who believes that Scratch does not help or hurt attitudes towards coding and computer science.

In a qualitative study by Fallon (2016) two elementary classrooms in a traditional educational setting underwent a study using Scratch Jr to see the effects on students’ Computational Thinking. One teacher had previous Scratch experience and the other did not, but none of the students in either class had any previous experience. Students worked in pairs to create a range of shapes and letters using Scratch Jr Thirty-two students or sixteen pairs of students participated in five sessions of twenty-five minutes to forty-five minutes, using Scratch Jr on iPads (Fallon, 2016). Fallon (2016) collected data using an embedded display and audio capture app installed on the iPads. Using Bloom's Taxonomy, the videos were watched, and data was coded from four hundred thirty-six events that occurred in the nine and a half hours of recorded data (Fallon, 2016). Forty-three of the events could not be agreed on to fit a category, so they were discarded. A limitation of this study is due to absences, student pairings occasionally were changed or altered.

Results from the Fallon (2016) study showed an improvement in understanding and clarifying steps and applying knowledge to test code. According to Resnick et al. (2009) and Fallon (2016), these are needed steps in Computational Thinking. Results also showed that some students were performing more advanced conceptual thinking such as using variables and sequencing. Additionally, in the Saez-Lopez et al. (2016) study, results showed significant improvement in learning computer coding concepts and Computational Thinking. Also, in the Papadakis et al. (2016) study there was an improvement in knowledge towards coding understanding and Computational Thinking. Papadakis et al. (2016) believes that scratch is better for younger students while AIA is better for older students because it uses their phone which is meaningful for youth. For optimal results Papadakis et al. (2016) believe students should be exposed to coding by using Scratch Jr first, then Scratch, then AIA, and finally a text-based computer coding language.

2.2 Theoretical Framework: The Nexus of Practice

The Knitting Code program aims to better understand if knitting can improve students' understanding of computer coding. Claims have been made that knitting is computer coding (Roberts, 2019). It is believed that the use of arts in STEM can promote a different type of learning due to the requirement to find a creative solution to a problem (Peppler & Wohlwend, 2018). The key is that art/craft must be included in a meaningful way that serves to improve learning through problem-solving and not just to make an item. Combining computer coding instruction with the arts has been a challenge in the past because computer coding has a history of being viewed as challenging to learn (Peppler & Wohlwend, 2018). By using craft/knitting, in combination with computer coding, the identities of both clash and produce a new learning experience that may enable females, minorities, and those not typically interested in coding to excel (Peppler & Wohlwend, 2018). Additionally, by combining computer coding and knitting the creative, problem-solving aspects of both may potentially improve learning.

The idea of combining different topics, such as knitting and computer coding, is referred to as the nexus of practice (Scollon, 2001). The nexus of practice, or more specifically, the nexus of STEAM, will be the guiding Theoretical Framework of the Knitting Code program. The nexus of STEAM believes that the use of arts in STEM can promote a different type of learning that is more meaningful and inclusive (Peppler & Wohlwend, 2018). By knitting and computer coding the identities of both clash and produce a new identity that may be more inclusive and provide different access points for those that struggle with traditional computer coding instruction (Peppler & Wohlwend, 2018).

The nexus of practice is a fairly new theory (Scollon, 2001). Books such as "Mediated Discourse" by Scollon (2001) and "The Nexus of Practice" by Hui, Schatzi, and Shove (2017) provide an understanding of the nexus of practice and the history. Research using the nexus of practice also serves to better explain the use of the theory. Research can be found using key terms such as nexus, practices, nexus of practice, and STEAM. The theoretical framework section is organized by first explaining the history of mediated discourse and the nexus of practice (Hui et al., 2017; Scollon, 2001), then research using the nexus of practice is discussed along with the similarities and differences on how the nexus of practice is used (Atalay, 2011; Palviainen, 2020; Wohlwend, 2008; Wohlwend & Medina, 2012). Next, the Nexus of Steam is discussed (Milroy, Holmes, & Wegener, 2015; Peppler & Wohlwend, 2018) along with the similarities and differences on how the nexus of practice is used (Gulhan & Sahin, 2018; Jawad et al., 2018; Peppler & Glosson, 2013; Rubio et al., 2014). The literature review concludes with the justification for using the nexus of practice in the Knitting Code program (Peppler & Wohlwend, 2018; Scollon, 2001).

2.2.1 History of Mediated Discourse Analysis and the Nexus of Practice

The nexus of STEAM (Peppler & Wohlwend, 2018) is the guiding theoretical framework for the Knitting Code program and is based on the nexus of practice (Scollon, 2001). Before the nexus of practice and nexus of STEAM can be understood, an understanding of mediated discourse analysis is needed. Mediated discourse analysis, proposed by Ron Scollon (2001), is an anthological study based on social action. Mediated discourse analysis evolved from critical discourse analysis by Chouliaraki and Fairclough (1999). The purpose of mediated discourse analysis is to "understand social life and social change” (Scollon, 2001, p. 8). Mediate is defined as an agreement, and discourse is defined as a written/spoken conversation/debate (Merriam-Webster.com, 2018). Mediated discourse is concerned with two questions, “what are the actions going on and how does discourse figure in these actions” (Scollon, 2001, p. 1). Scollon (2001) argues that mediated discourse analysis has ties with cognitive development theory by Vygotsky, because “learning proceeds from social interaction through processes of social interaction to the reproduction on the instrumental plane of human psychological structures” (p. 9). Scollon (2001) explains mediated discourse through the action of ordering a cup of coffee with friends.

A simple invitation to go get a cup of coffee is more complex than it appears. Going for a cup of coffee can be viewed as a single action of getting a cup of coffee; or the invitation to get a cup of coffee can be viewed as a set of multiple actions such as driving to the coffee shop, drinking coffee, talking, and throwing away trash (Scollon, 2001). Additionally, when looking through the scope of discourse, the action of getting a cup of coffee could be considered to have one discourse, having a conversing with friends while drinking coffee, or a set of multiple discourses, marketing of coffee, discussing the order, talking about the family, etc. (Scollon, 2001). Additionally, when ordering a cup of coffee, the cup is the most important element of having a cup of coffee. Without the cup, coffee cannot be given and consumed by the individual (Scollon, 2001). The cup is the "material line that holds it all together" (Scollon, 2001, p. 2). The coffee cup itself is comprised of discourses as well including the branding of the logo on the cup, legally registering the logo as that of the company, a website listed on the cup, a statement on the cup explaining their view of coffee growing ethics, information on what the cup is made of/if it can be recycled, and finally there is manufacturing information (Scollon, 2001).

Having a cup of coffee is more complex than it initially seems. Ordering a cup of coffee is a combination of action (simple or complex) and discourse (simple or complex). Mediated discourse looks at the relationship between the action and discourse without giving either one too much attention over the other (Scollon, 2001). Additionally, mediated discourse is comprised of five concepts: “mediated action, site of engagement, mediated means, practice, and nexus of practice” (Scollon, 2001, p. 3).

These five concepts work together to clarify mediated discourse. Mediated action means that the action, not discourse, is the unit of analysis for mediated discourse, because, without the action, the discourse would not occur (Scollon, 2001). Additionally, Scollon (2001) states that an action needs to be materially grounded, not an abstract idea. In the coffee example, the mediated action would be the act of going to go get coffee. “There can be no action without participation in such discourse; no such discourses without concrete, material actions” (Scollon, 2001, p. 3). The site of engagement is the moment in time when an action occurs (Scollon, 2001). The action is unique to that certain time, and that time needs to be observed when analyzing the action. In the coffee example, it would be the date, time, and location that the individual is going to go get coffee. Mediated means is how the action is carried out through material objects (Scollon, 2001). In the coffee example, this would be the coffee cup. Practice is the social practices and “a mediated action is only interpretable within practices” (Scollon, 2001, p. 4). In the coffee example, the practice would differ based on where the coffee was made such as a restaurant of a coffee shop. Finally, the nexus of practice is how practices are linked to each other. In the coffee example, the actual ordering is the practice, because the individual must identify what they want and the size (Scollon, 2001). When combining these five components they form a "constellation of linked practices which makes for the uniqueness of the site of engagement and the identities thus produced, not necessarily the specific practices and actions themselves" (Scollon, 2001, p. 5).

Now that mediated discourse analysis has been described, the nexus of practice can be better explained. Giddens defines a practice as an organized set of actions (1984, as cited in Hui et al., 2017), and social practice is a practice that has been developed through mediated actions until it becomes a norm or habit (Scollon, 2001). As mentioned before, the nexus of practice is how practices are linked to each other, and Scollon (2001) describes the nexus of practice, once again, through the action of ordering a cup of coffee. Scollon (2001) explains that he had practiced ordering coffee before, but the site of engagement differed so his previous practice was not helpful. As a result, Scollon (2001) used the menu to improve his knowledge of ordering a cup of coffee, but when he asked what the difference was between a latte and café au lait when ordering it marked him as unfamiliar with the practice. He marked himself "as an outsider, but wishing to come in" (Scollon, 2001, p. 141). Poor performance in a practice signals alienation of a group, while the successful performance of action signals membership of the group (Scollon, 2001). The idea from this is that Scollon (2001) was trying to form a membership or an identity with those that drink coffee at this shop through practice. As Scollon (2001) explains "one's actions produce one in the first place as a person who is competent or not in some social practice, and in the second place, they produce one as someone with an identity" (p. 142). The nexus of practice argues that when social practices overlap, although they are never finalized, the practices form a web that signals identity (Scollon, 2001).

The nexus of practice evolved from the idea of a community of practice developed by Bourdieu in 1977. After its initial introduction, the community of practice was reworked and reintroduced in 1990 (Scollon, 2001). Bourdieu (1977, as cited in Scollon, 2001) argues in the community of practice people are members of the same group based on their habits, people cannot be members of several groups, and people do not move from one group to another. Once a person is a member of a group they are bound and cannot overlap into another group. Scollon's (2001) theory of nexus differs from this because Scollon believes that practices determine identity and acceptance into a group, but social practices are never finalized and often intersect with other social practices. Nexus refers to, all the practices. For example, teaching is a nexus. Practice refers to all the components that build the nexus. For example, practices that fall under teaching include creating assignments, grading assignments, creating assessments, communicating with parents, etc. (Scollon, 2001).

The nexus of practice can now be understood as the combination of different practices that form a nexus through links and relationships (Hui et al., 2017). As people develop practices, they develop certain abilities that signify membership in a group (Hui et al., 2017). Practices can become organized into a nexus through multiple methods including gathering around a place or time (Hui et al., 2017). Learning, both passive and active, is necessary to form practices, and as a result signals membership to a group (Hui et al., 2017). Additionally, learning occurs when different nexus intersect, because knowledge is shared through the interaction (Hui et al., 2017). Conflicts can also occur when nexus converge because they may not share practices, and as a result, old practices can develop into new practices (Wohlwend, 2014). The new practices are not what form the change, instead how the practices interconnect and display a new organization of practices is what causes a change (Hui et al., 2017). This is referred to as a disruption of practice (Scollon, 2001). Key practices from the different nexus may form a new nexus and provide an invitation for new membership of that nexus that combines key practices and values from the combined nexus (Medina & Wohlwend, 2014). When nexus combine, the new nexus can change practices and identities that would be slow to change (Wohlwend, 2014). “When new practices emerge in nexus, the results can be transformative, allowing greater access and broader participation” (Peppler & Wohlwend, 2018, p. 91).

Materials theory also plays a role in the nexus of practice. Within a nexus and the practices, knowledge, materials, and tools are used (Holland & Cole, 1995). The practice determines the expectation of knowledge, materials, and tools. This creates expectations of what the items are used for and who is allowed to use them (Holland & Cole, 1995). “Each cultural practice—with its related tools and materials— conveys distinct expectations for who and what constitutes experts and expertise” (Peppler & Wohlwend, 2018, p. 91). For example, someone that belongs to the nexus of construction would have the expectation of knowing how to use a saw, while someone that belongs to the nexus of sewing would have the expectation of knowing how to use a sewing machine. The expectation can also go the other way. For example, if someone knows how to create a circuit board, they would be expected to be an electrical engineer. Additionally, tools and materials often have associations in history and culture (Butler, 1990). This, as a result, leads to a different association between males and females (Peppler & Wohlwend, 2018). This means, based on historic norms and practices, some items, such as a power tool, are considered more masculine, while some tools, such as finger-nail polish, are considered more female (Peppler & Wohlwend, 2018). To disrupt these stereotypes of who uses what, a disruption in nexus needs to occur (Scollon, 2001).

The nexus of practice does have some limitations. First, while there are claims about the effects of the disruption of nexus there is not much evidence to back the claim. There is evidence about the nexus of practice, but not on what happens when nexus are combined (Peppler & Wohlwend, 2018). The evidence may exist, but not using the terminology of nexus or practice. Additionally, since the introduction to the materials theory, gender neutrality and gender roles have changed. Historically, an item may have held a feminine or masculine identity, but today that is not considered socially acceptable (Peppler & Wohlwend, 2018). More research is needed on the materials theory in modern society.

2.2.2 Studies Based on the Nexus of Practice

The nexus of practice examines what happens when different nexus are combined (Scollon, 2001). When an overlap occurs from different nexus new learning and membership are possible (Hui et al., 2017). The nexus of practice is part of mediated discourse analysis that looks at the interactions of actions and discourses that occur (Scollon, 2001). The nexus of practice has been used to better understanding multiple types of interactions of nexus.

In a study by Wohlwend (2008), kindergarten, as a nexus, was studied using reading, writing, play, and design as the practices. Wohlwend (2008) investigated how play practices and design practices impact an understanding of early literacy; and, how the nexus of play, design, and writing impact diverse learners. Data was collected through teacher interviews, classroom visits, and classroom environment surveys. The study occurred in an all-day traditional classroom setting that contained twenty-one students from diverse backgrounds. Mediated discourse analysis and multimodal analysis were used to gain a better understanding of the results (Wohlwend, 2008). Results included students "playing" as a teacher and reading to each other, students authoring books, and using literature to reenacting sports events (Wohlwend, 2008). By combining the practices of reading, writing, play, and design students were able to make sense of their learning. Wohlwend (2008) states by combining the nexus of school and the nexus of play and design students were able to improve their literacy.

Additionally, in a study by Palviainen (2020) the nexus of video and communication, of two multilingual families, was studied. Both families consisted of a single mother in her thirties with a four-year-old child. In modern society, a variety of video communication software has developed including Skype, Zoom, and Google Hangouts. This article questions if adding the nexus of video to the nexus of communication impacts the emotional relationships of these mothers to a significant other living abroad (Palviainen, 2020). “The daily video calls are seen as a nexus of practice, i.e., a constellation of social, linked practices. The overall aim of the study is to navigate the nexus of practice, i.e., to identify linked and recurring practices in the video calls across the two trans-local family configurations” (Palviainen, 2020, p. 86).

Additionally, the researcher studied how the mediated action of using technology impacts communication. Data was collected in three stages including researcher-led interviews, participant-led interviews, and recall interviews. Additionally, one mother made notes on an iPad about her video calls and the other mother recorded some of her video phone calls (Palviainen, 2020). The researcher used a three-step method to analyze the nexus of practice including engaging the nexus of practice (social actions identified), navigating the nexus of practice (map discourses), and identifying changes in the nexus of practice (Palviainen, 2020). Results showed that by combining video with communication enabled emotional bonds (Palviainen, 2020). Additionally, Palviainen (2020) argues that digital video practices aid in the development of language.

Finally, in a study by Wohlwend and Medina (2012) the nexus of media and education was studied through the show "What Not to Wear." The study questioned how the overlap between media and education impacts identity revision (Wohlwend & Medina, 2012). This research arose out of the idea that media impacts expectations of gender models and racial representations that may affect self-image (Wohlwend & Medina, 2012). The show "What Not to Wear" uses items and clothes to develop an individual's identity. Wohlwend and Medina (2012) "examined how this fashion makeover program teaches participants and viewers to value dominant gender and ethnicity performances through negative fashion readings of (primarily) women's bodies" (p. 547). Mediated discourse analysis was used to understand the overlap of the different nexus (Wohlwend & Medina, 2012). First, Wohlwend and Medina (2012) engaged the nexus of practice by identifying the main actors, practices, and transformative events. Second, Wohlwend and Medina (2012) navigated the nexus of practice by identifying reoccurring types of scenes and types of clothes. In the third step, Wohlwend and Medina (2012) identified the nexus by determining how the practices of media education impact belonging. The results of the study showed that "first, the program is a dramatization that represents one woman's (portrayal of) lived practices and clothing choices which are read on her body as a personal expression of fashion trends. Second, each videotaped episode in the reality program is a globalized lesson, situated in the nexus of discourses about gender, ethnicity, and consumerism that shape viewer identities" (Wohlwend & Medina, 2012, p. 557). Wohlwend and Medina (2012) argue that media, when combined with education/instruction, can have transformative power.

2.2.3 Similarities and Differences Between Nexus of Practice Studies

All three studies focused on different topics but were guided by the nexus of practice (Palviainen, 2020; Wohlwend, 2008; Wohlwend & Medina, 2012). There were differences between the studies. First, all three studies used the nexus of practice to look at very different social interactions. In the study by Wohlwend (2008) interactions between children in a classroom were studied. In the study by Palviainen (2020) interactions between a mother, her child, and another individual on a video class were observed. Finally, in the study by Wohlwend and Medina (2012) interactions that occurred on a television program were observed.

Second, the nexus of practice in the three studies were used for different purposes. In the first study Wohlwend (2008) examined the nexus of kindergarten and how the practices (reading, writing, play, and design) impact learning. In this study, there was a slight study of the disruption of nexus when play and literacy overlapped. Additionally, there was mention of materials theory, (Butler, 1990) in that the boys preferred to apply sports (viewed as masculine) to literacy while the girls preferred to play school (viewed as feminine) to improve their literacy. Overall, this study showed that the disruptions improved learning and that play needs to be a part of kindergarten, but the author did not examine the impact of material theory on learning. Palviainen (2020) used the nexus of practice for a very different reason. In this study, no disruption occurred, but different practices of communication were observed including audio, video, and language. Palviainen (2020) determined that the practice of video communication did improve the overall nexus of communication by improving the emotional connection. The study by Wohlwend and Medina (2012) once again looked at the nexus of media and how an educational program such as "What Not to Wear" impact identity. While the main purpose of the show may not be to educate what people should wear it, does have an adverse effect. The show frequently discusses "correct" practices. By the show explaining, showing, and modeling how one should look, a viewer will either identify or change their identity to identify with the practices of the show. Wohlwend and Medina (2012) wanted to show how media along with magazines, movies, posters, etc. impact a person's identity.

While these studies differed, they were also similar. All three studies examined forms of human interaction. The social interaction was the focus, and the nexus of practice served to explain the interaction (Palviainen, 2020; Wohlwend, 2008; Wohlwend & Medina, 2012). While the interactions were in different forms there was still a mediation to form a discourse (Scollon, 2001). This mediation included a combination of skills that determined an identity. The cumulative term for the skills in a nexus (Scollon, 2001).

Additionally, in all three studies, the researchers were using the nexus of practice to explain why people interact the way they did and what shaped their desire to identify with a nexus (Palviainen, 2020; Wohlwend, 2008; Wohlwend & Medina, 2012). In the study by Palviainen (2020) and Wohlwend and Medina (2012) the formation of identity impacted either what language was spoken or what clothes to wear. In both studies, mediated discourse analysis was used to engage, navigate, and identify practices that determined identity. Materials also played a role when explaining social interaction through the nexus of practice. In the study by Wohlwend (2008) and Wohlwend and Medina (2012) materials impacted identity. The nexus of practice was used to explain this phenomenon. These items historically had a stigma and did impact the practice.

The nexus of practice is important for the Knitting Code program because different nexus (knitting and coding) will be combined. Where the practices overlap is where the Knitting Code program will emerge from. To gain a better understanding, the nexus of art and STEM need to be examined.

2.2.4 The Nexus of STEAM

Peppler and Wohlwend (2018) conducted research on combining art and STEM. The research is based on the nexus of practice, but Peppler and Wohlwend (2018) developed their own term, nexus of STEAM, to explain their findings. In this adaptation of the nexus theory, the nexus of art and the nexus of a STEM are combined into a new nexus that uses a combination of practices from both nexus (Peppler & Wohlwend, 2018). The combination creates a disruption of nexus. The results of this new nexus can provide different acceptance points of membership and identity that may have previously been blocked due to not fitting in with the original practices (Scollon, 2001). Therefore, by combining the nexus of art and the nexus of STEM, the new nexus of STEAM may enable participation, leadership, and access that was not there before (Peppler & Wohlwend, 2018). Combining the nexus of art and STEM has been occurring for years (Atalay, 2011; Peppler & Wohlwend, 2018). “Art and science has a long, entwined history dating back to Plato and Aristotle, through da Vinci and later Enlightenment...In the 21st century, attention has turned one again to the intersections of art and science, at both theoretical and practical levels” (Peppler & Wohlwend, 2018, p. 90). For example, Leonardo da Vinci drew some of the first realistic images of human anatomy which is considered a great work of art, yet his motivation stemmed from science and the desire to understand the human body (Atalay, 2011). Additionally, several current artists either collaborate with individuals in different content or have cross-disciplinary training themselves that enables the artist to combine art with another nexus (Milroy et al., 2015; Peppler & Wohlwend, 2018).

When the nexus of steam occurs two limitations do occur. Fist, when one nexus (i.e., crochet) is combined with another nexus (geometry) the new nexus will only contain some of the aspects from each nexus. That means that the new nexus will not be as in-depth about either of the original nexus, but the new nexus will provide a point of entry that will enable the individual to cross over to the original nexus (Peppler & Wohlwend, 2018). When combining nexus, the individual needs to determine how in-depth they want to go. If a basic understanding is all that is needed, then the new nexus will serve. If the individual wants to go more in-depth, then they can use the practices learned in the new nexus to build their understanding of the original nexus. The second limitation is that the combination of art and STEM must be combined in a meaningful way that serves to improve learning (Peppler & Wohlwend, 2018). The practices from both nexuses must be combined in a way the enables the learner to understand more about both nexus. In other words, an instructor cannot just ask the student to make a model of a cell and expect the nexus of steam to occur. Instead, practices from both contents (i.e., functions of organelle and scale) need to be used to deepen the understand and make the combination of nexus meaningful. Focusing on one nexus without giving the other nexus equal thought will not result in a nexus of STEAM.

2.2.5 Studies Based on the Nexus of STEAM

By combining the Nexus of art and science, modes of learning and knowing were challenged by concepts of languages and materials from each discipline (Milroy et al., 2015). While the original goals of art and science are different there is a deeper understanding when they are combined due to the processes, motivations, and curiosity (Milroy et al., 2015). Both science and art search for meaning, which is their commonality, even though they appear to have different goals at first (Milroy et al., 2015). "In every era, the arts reflect the current historical movement...today one of the most pervasive impacts on the world around is arguably the influx of new technologies, so it should come as no surprise that new technologies are being used ubiquitously as creative tools in the arts" (Peppler & Wohlwend, 2018, p. 89). Even though this is the case, learning to code has been met with the most hesitance from art educators due to the perceived lack of expressive medium in coding. Additionally, STEM has struggled to bring new people into computer science due to the barrier of learning to program (Peppler & Wohlwend, 2018, p. 92). This is why a disruption of Nexus was needed, for both, to attract a new audience.

In a study by Rubio et al. (2014), the idea that perceptions of computer coding would alter when combined with a different nexus was studied. In this instance, a college biology class was used because biology is often viewed as more gender-neutral or even slightly more female-dominated (Rubio et al., 2014; Starr, 2018). In this study seventy-six, college freshmen with no previous computer coding experience were used (Rubio et al., 2014). Of the seventy-six students, half were enrolled in a traditional computer coding class (control group) and the other were enrolled in a biology course that incorporates computing, biology, and art (experimental group). Physical computing is similar to "unplugged" coding because it removes computational concepts out of the screen and into the real world so that students can interact with them (Rubio et al., 2014). During the study students' perceptions of coding and learning were monitored through assessments and compared using a t-test (Rubio et al., 2014). After the study, Rubio et al. (2014) discovered that the gender gap difference of those interested in coding when comparing the control and experimental group was non-existent. By combining coding and another subject, similar to combining coding with knitting, the views of coding were shifted, and the new modules proved more effective in changing perceptions of coding. Additionally, those that participated in the physical computing/biology coding class also showed a greater understanding of coding (Rubio et al., 2014). Although the sample size was low, this study serves as a path to question how combining computer coding with other courses will impact those that usually wouldn't be interested in computer coding and their level of understanding.

In another study by Gulhan and Sahin (2018) the value of adding art as a design aspect into STEM was examined. Thirty students in the seventh grade participated in five

STEAM activities on the reflection and absorption of light. The lesson took five weeks and a total of twenty hours (Gulhan & Sahin, 2018). After the activity, Gulhan and Sahin (2018) examined projects that combined elements of the engineering design, process, art, and science and saw positive results that combined creative design that was linked with the science content. Six students were also interviewed after the activity. Gulhan and Sahin (2018) asked students if adding arts increased their interest in STEM fields, and students said yes because art helped them understand science concepts better and the communication and teamwork in the design process was useful. Finally, when Gulhan and Sahin (2018) asked which of the fields in STEAM did students enjoy the most, four of the six students replied that art was their favorite field. By combining art, a new group of students were brought into STEM.

In a third study by Jawad et al. (2018) the effect of integrating art and animation in teaching computer coding to high school students was explored. The study focused on high school students' interest and knowledge of computer coding, as well as the students' interest in pursuing a degree in computer science after graduation (Jawad et al., 2018). Data was gathered through pre- and post-test surveys using a five-point Likert scale to measure students' computer science degree interest (Jawad et al., 2018). Jawad et al. (2018) determined through the study that by combining art with science students' knowledge, enjoyment, motivation in learning computer coding, and their interest in pursuing a degree in computer science after graduation increased. What was surprising is that there was a greater in females than males because a new approach combining art and science was used (Jawad et al., 2018).

Finally, the implementation of the Nexus theory can also be seen in a study by Peppler and Glosson (2013) who examined the use of an e-textile toolkit and if it could aid youth in learning about electronics in an elective setting; as well as whether e-textiles could elucidate important circuitry concepts that traditional materials have historically struggled to convey. The research had two goals: to determine how youth learn about electrical circuits in an elective environment, and how can e-textiles facilitate the learning of important concepts in electrical circuits that traditional materials have historically struggled to convey (Peppler & Glosson, 2013). Overall, Peppler and Glosson (2013) tested whether youth could create an overall working circuit by whether they understood three core concepts: current flow, connections, and polarity by combining art and science in a combination of the Nexus theory. The study by Peppler and Glosson (2013) focused on seventeen youth between the ages of seven and twelve at a local Boys and Girls Club. The twenty-hour after school program that met two times a week for two hours each was designed to teach the youth about electrical circuitry and then design an e-textile project combining the art of the e-textile with the science of the circuitry. Peppler and Glosson (2013) collected data through pre- and post-tests to gauge students' understanding of current flow, connection, and polarity. Afterward, data was analyzed using a sequential mixed-methods approach (pre-post with paired samples, videotaped observations, artifacts, circuit design assessment) and the circuit design assessment was analyzed by coding the youths' responses on a five-point scale (Peppler & Glosson, 2013). Materials that were used included LEDs, Lily Pad button boards, coin cell batteries plus holders, conductive thread, in addition to several textile and craft materials (Peppler & Glosson, 2013).

Peppler and Glosson's (2013) results revealed that the Nexus of art and science did produce positive results including that youth were engaged in the e-textile design, demonstrated gains in their ability to produce a model drawing of a working circuit, and could explain how current flow, polarity, and connections worked. What was interesting in this study was that the girls took leadership roles when the activity revolved around sewing, even when the boys had more experience with sewing (Peppler & Wohlwend, 2018). The nexus of art or science determined who took the lead on the activity based on the practice (and gendered history). Girls were placed in leadership roles when it was time to sew or craft and boys placed in leadership roles when it was time to test or solder (Peppler & Glosson, 2013; Peppler & Wohlwend, 2018). This is interesting because the females showed more leadership and confidence when using sewing even if their male counterpart was more experienced in sewing. By combing the two nexus, females felt the confidence to take a chance and use their new knowledge. Additionally, professional e-textiles are becoming the first-ever female-dominated computer community. "With the intersection of the arts (crafts), there is a shift in masculine dominated practices typically found in STEM culture" (Peppler & Wohlwend, 2018, p. 95).

2.2.6 Similarities and Differences Between the Nexus of STEAM Studies

All of the above studies focused on different topics but were guided by the nexus of STEAM. While all the studies focused on the nexus of STEAM, there were differences. In terms of social interactions, there were a variety of ages and classroom settings. The studies ranged from college-age students in a biology classroom setting (Rubio et al., 2014) to elementary age students in an after-school outreach program (Peppler & Glosson, 2013). How the nexus of STEAM was used also differed between the studies. In all four studies, the art and STEM differed. The types of art and STEM ranged from using animation to teach computer coding (Jawad et al., 2018) to using light art to teach light waves (Jawad et al., 2018).

While the studies themselves differed, there were similarities. First, in all four studies, social interactions were studied. In each study, a STEM classroom was used, and art was being added compared to an art classroom with science being added. Additionally, art was being used to improve the interaction between students and material. Second, in all four studies art was combined with STEM to improve understanding and identity. In each study, students reported an improved understanding and enjoyment of content when combined with art (Gulhan & Sahin, 2018; Jawad et al., 2018; Peppler & Glosson, 2013; Rubio et al., 2014). Additionally, Rubio et al. (2014), Jawad et al. (2018), and Peppler and Glosson (2013) reported gender changes when the nexus of steam occurred. Females started to perform at the same level as males (Rubio et al., 2014), more females reported they were interested in pursuing a career in STEM (Jawad et al., 2018) and more females took on leadership roles (Peppler & Glosson, 2013).

The promising results of combining art with STEM have provided the foundation for the Knitting Code program (Gulhan & Sahin, 2018; Jawad et al., 2018; Peppler & Glosson, 2013; Rubio et al., 2014). While the nexus of STEAM is a term created by Peppler and Glosson (2013) it is based on the nexus of practice (Hui et al., 2017; Scollon, 2001). Therefore, the nexus of practice, and more specifically the conflict of nexus, will be the guiding theoretical framework for the Knitting Code program.

2.2.7 Justification of Knitting Code

The Knitting Code program has several focuses. First, the Knitting Code program wants to provide more accessibility for those that would not typically computer code. Second, the Knitting Code program aims to improve the understanding of computer coding by improving computational and Algorithmic Thinking. It is believed that by improving these skills at a young age students have a greater ability and chance to pursue computer science in the future (Sung, 2019). Finally, the Knitting Code program wants to investigate how combining codding plugged (Resnick et al., 2009) and coding “unplugged” (Bell & Vahrenhold, 2018; Lee & Junoh, 2019; Tonbuloglu & Tonbuloglu, 2019) by using knitting impacts students understanding of coding.

The Nexus Theory of combining art/craft (knitting) and computer science (coding) is the guiding framework of the "Knitting Code" program. By combining different Nexus, a new Nexus can be made. This new Nexus can make content more approachable for some and diminish stereotypes that are preventing others from identifying with computer science. The Nexus Theory is appropriate for several reasons.

First, the Nexus theory can change the stereotypes of who codes. By combining a Nexus that may not normally "fit" into computer science a disruption occurs. This disruption allows those that don't normally identify with the stereotype to relate possibly now. This is a guiding component of Knitting Code because the program allows those that associate with craft or art to associate with coding. A survey will be used to monitor students' identity as they go through the program to understand if knitting can disrupt the stereotype of a computer scientist.

As mentioned before, identity plays a role in who takes an interest in learning to code and majoring in computer science (Ehrlinger et al., 2018). Women's interest in computer science might be encouraged through interventions that combat more extreme stereotype-based perceptions such as combing the Nexus of computer science with the Nexus of art (knitting), instead of, or in addition to, trying to change women's self-views (Rubio et al., 2014). This way the woman can develop a new view of who participates in computer science.

By combining two different Nexus to create a new one the natural identity of that field may change and make it more accessible for those it previously was not such as females, minorities, and those not traditionally interested in STEM (Peppler & Wohlwend, 2018). This contrasts with the idea that these groups lack the skills or are not able to be successful in computer science (Peppler & Wohlwend, 2018). Scollon (2001) believes that by making a change on how STEM is approached, such as sewing and crafting, can create meaningful differences in access, participation, and leadership (Scollon, 2001).

Second, the Nexus Theory promotes computational and Algorithmic Thinking by focusing on developing steps to solve a problem and use critical thinking skills. Incorporating these thinking skills will be the guide when designing the curriculum for both the comparative and experimental group. Students will be asked to create decompositions of everyday activities or simple stories at the start of the program (Lee & Junoh, 2019). As the program advances students may find it easier to apply Computational Thinking and Algorithmic Thinking to either computer coding or knitting. This way, once the nexus are combined, students can carry over what they used from one discipline to the other. As a result, the idea of breaking down the problem and creating an algorithm to solve will be applied to both computer coding and knitting. Additionally, each of these skills will demonstrate and possibly improve these skills that are necessary to learn to code.

There are several reasons that combining the nexus of art with the nexus of computer coding will improve learning. First, when combining the nexus of art and science there can be an improvement of technical and observational skills. This can improve creative and critical thinking, which are components of computational and Algorithmic Thinking (Adkins et al., 2017). Students will be able to use what they learn from one discipline and apply it to the other. This may allow the participant to look at the problem from a different perspective. For example, if students are struggling with understanding how to program a loop, they may be able to write out a knitting pattern to demonstrate their understanding. Then, the students can use this knowledge to approach the computer coding question again while using their new understanding of the solution.

Second, Computational Thinking and Algorithmic Thinking do not require the use of a computer. A computer may become a distraction to developing skills (Yadav, Zhou, Mayfield, Hambrusch, & Korb, 2011). By using knitting, Algorithmic Thinking and Computational Thinking skills can be developed unhindered. Then, when the nexus of knitting is combined with the nexus of coding these learned skills can be carried over. For example, learning to write an algorithm for a certain specialty knit stitch, and then applying algorithm writing to a computer coding problem.

Third combining knitting and coding can improve thinking skills. Key elements of Computational Thinking include (1) abstraction and automation (2) systematic process and information (3) symbol systems and representations (4) algorithmic notions of flow control (5) structured problem decomposition (6) iterative, recursive, and parallel thinking (7) conditional logic (8) Efficiency and performance constraints (9) Debugging and systematic error detection (Grover & Pea, 2013). Algorithmic Thinking includes (1) decomposition (2) repetition (3) data organization (4) generalization (5) parameterization (6) algorithm vs. program, top-down design (7) refinement (8) critical thinking (Cooper et al., 2000; Hurlburt, 2018). All these skills are needed to become proficient coders, but knitting can be a means to teach these skills.

Finally, the Knitting Code program will combine a variety of tools including STEAM, block-based computer coding, and “unplugged” coding. STEAM is a combination of art and STEM. This is an example of the nexus theory because there is a combination of contradictory fields. By combining both, the learner will be challenged to use creative design elements to solve problems. Second, the "Knitting Code" program will use drag-and-drop programs so that the material is accessible to younger students. The goal of drag-and-drop is to introduce students to computer coding in a friendly manner. Drag-and-drop will combine the aspects of art and computer coding. “unplugged” coding will also be used through knitting. This “unplugged” component will allow students to learn to knit and then knit their computer code. Since “unplugged” coding was never intended to be taught independently, the lessons will be combined with a drag-and-drop coding-based curriculum.

The Nexus theory can benefit how computer coding is taught through STEAM, block-based coding such as Scratch, and “unplugged” strategies. The very nature of the nexus theory is combining two seemingly unrelated nexus to create a new nexus. STEAM is an example of this because art is being combined with STEM to create something new. Additionally, the nexus theory believes that Scratch, a block-based program, is so impactful is because it blends computer coding, media, and arts working together to transform what it means to engage in each of these disciplines (Kafai & Peppler, 2011). Additionally, using "unplugged" computer science allows for the combination of art and computer coding (Bell & Vahrenhold, 2018). By using the "unplugged" method barriers such as a weak background in math can be overcome by fusing the instruction with different content that the student feels more confident in.

The focus of the Knitting Code program is to understand how the combination of two different nexus will combine. The study hopes that the results will show improvement in understanding and provide a door of accessibility for those that would not normally be interested in computer coding. There is not much research on the conflict of nexus (Scollon, 2001), even though there are some hypothetical explanations. There is research on combining the nexus of art and STEM (Peppler & Wohlwend, 2018), but the studies are small. More research is needed in understanding how combining knitting and computer coding will impact students and the nexus of practice, or more specifically the nexus of STEAM, will be the guiding theoretical framework to explain the social and material practices.

2.3 Summary

In conclusion, the Nexus Theory combining art/craft (knitting) and computer science (coding) is the guiding framework of the "Knitting Code" program. By combining different nexus, a new Nexus can be made. This new nexus can make content more approachable for some and diminish stereotypes that are preventing other from identifying with computer science. The nexus theory is appropriate for several reasons based on the themes discussed in the literature review.

First, the nexus theory can change the stereotypes of who codes. By combining a Nexus that may not normally “fit” into computer science a disruption occurs. This disruption allows those that don’t normally identify with the stereotype to relate possibly now. This is a guiding component of “Knitting Code,” because the program allows those that associate with craft or art to associate with computer coding. A survey will be used to monitor students’ identity as they go through the program to understand if knitting can disrupt the stereotype of a computer scientist.

Second, the Nexus Theory promotes computational and Algorithmic Thinking by focusing on developing steps to solve a problem and use critical thinking skills. Incorporating these thinking skills will be the guide when designing the curriculum for both the comparative and experimental group. Students will be asked to create decompositions of everyday activities or simple stories at the start of the program (Lee & Junoh, 2019). As the program advances students may find it easier to apply Computational Thinking and Algorithmic Thinking to either coding or knitting. This way, once the nexus are combined, students can carry over what they used from one discipline to the other. As a result, the idea of breaking down the problem and creating an algorithm to solve will be applied to both coding and knitting. Additionally, each of these skills will demonstrate and possibly improve these skills that are necessary to learn computer coding.

Finally, the “Knitting Code” program will combine a variety of tools including STEAM, block-based computer coding, and “unplugged” coding. STEAM is the combination of art and STEM. This is an example of the nexus theory because there is a combination of contradictory fields. By combining both, the learner will be challenged to use creative design elements to solve problems. Second, the “Knitting Code” program will use drag-and-drop programs so that the material is accessible to younger students. The goal of drag-and-drop is to introduce students to coding in a friendly manner. Drag-and-drop will combine the aspects of art and computer coding. “unplugged” coding will also be used through knitting. This “unplugged” component will allow students to learn to knit and then knit their code. Since “unplugged” coding was never intended to be taught independently, the lessons will be combined with a drag-and-drop coding-based curriculum.

In conclusion, the nexus theory is guiding this research due to the combination of art and coding. The different components found through the literature review have shaped the goals of the “Knitting Code” program.

Chapter 3 METHODOLOGY

3.1 Introduction

Cases studies examine modern phenomenon in a current societal context. This is especially true of the relationship between the phenomenon and context is not clear. Case studies naturally have more variables than data, so as a result case studies require a variety of types of evidence to form triangulation and the use of previous theoretical ideas to guide data collection. (Yin, 2013).

The purpose of this case study was to determine if knitting could help teach Computational Thinking skills and to determine how identity was impacted by combining knitting and computer coding. A comparative group that incorporates computer coding and coding "unplugged" was compared against an experimental group that incorporates computer coding and knitting instruction. The purpose of this comparison was to see if incorporating knitting produces similar results. The results were similar between the "Knitting Code" experimental group and comparative, so the statement that knitting can teach coding can be confirmed within the defined parameters of the case. If the results had showed that the "Knitting Code" experimental group was not as successful as the comparative group, then the statement that knitting should not be used to teach coding would have been confirmed within the defined parameters.

Based on results from the literature review several decisions were made about the design of this study. First, the field of computer coding served as the main content with knitting being added, compared to knitting being the main content with computer coding being added (Adkins et al., 2017; Gulhan & Sahin, 2018; Jawad et al., 2018; Looi et al., 2018; Peppler & Glosson, 2013; Thuneberg et al., 2017). Second, a pre- and post-survey/questionnaire were used (Adkins et al., 2017; Jawad et al., 2018; Looi et al., 2018; Peppler & Glosson, 2013; Thuneberg et al., 2017). Additionally, the end of unit project was used (Adkins et al., 2017; Gulhan & Sahin, 2018; Jawad et al., 2018; Peppler & Glosson, 2013; Thuneberg et al., 2017). Finally, video journals, interviews, and physical journals were be used (Adkins et al., 2017; Gulhan & Sahin, 2018; Jawad et al., 2018; Looi et al., 2018; Peppler & Glosson, 2013; Thuneberg et al., 2017).

This case study was considered an exploratory case study since the goal was develop a hypothesis and justify further research (Yin, 2013, p. 6). Research questions are highly important in exploratory case studies because they define the who, what, when, where, how, and why (Yin, 7). This study focused on answering two research questions:

- Research question 1: How does learning to knit while learning to computer code facilitate the understanding of Computational Thinking skills including abstraction, Algorithmic Thinking, and understanding of parallelization in adolescent students?
- Research Question 2: How does combining knitting and computer coding impact the identity of who studies computer science in adolescent students?

Due to nature of observations, qualitative data was used for this case study. The data was deductively coded. There is not one best way to code because it relies on individualistic interpretation, so consistency was important (Saldana, 2021, p. 4). To be consistent, data was analyzed using QDA Miner Lite to organize and code data. This also helped to eliminate bias since the qualitative data has a set of parameters when being coded (Saldana, 2021). “Code” is a word of phrase that was assigned to data in a variety of formats that summarizes and represents findings (Saldana, 2021, p. 5). This code is not the same as computer coding mentioned earlier. These data codes helped take the data and make meaning out of what was collected. There are two phases of coding. “First cycle coding is analysis - taking things apart. Second cycle coding is synthesis – putting things together into new assemblages of meaning” (Saldana, 2021, p. 6). In this chapter the study design and data collection will be discussed.

3.2 Explanation of Case Study

3.2.1 Purpose of a Case Study

The main purpose of a case study is to answer the “how” and “why” questions, when conditions are hard to control, and the study focuses on modern phenomenon in a real-life context (Yin, 2003, p. 1). A case study can be used to increase understanding about the relationship between an individual, group, organization, social situation, or political setting and a phenomenon (Yin, 2003). Overall, a case study is a form of social science that strives to understand a “complex social phenomenon” (Yin, 2003, p. 2). Due to this most case studies are explanatory.

The number of cases is not as important as other types of studies because case studies are more interested in how the cases are impacted over time and less concerned with the frequencies or incidents that other types of studies would focus on (Yin, 2003). Case studies can deal with a variety of types of evidence including documents, artifacts, interviews, and observations (Yin, 2003, p. 8). The end purpose of a case study is to make a generalized statement regarding a theoretical idea or phenomena, not to make statements about individuals or populations. In other words, the main focus of a case study is dig deeper into theories, to expand findings, and to develop generalized theories, not to make statistical generalizations that would be more common in other types of experimental design (Yin, 2003, p. 10).

3.2.2 Reasoning Why a Case Study Was Used

A case study was determined to be the best approach for this study because the researcher was seeking to better understand how cases that underwent an unusual situation of learning knitting while also learning computer coding were impacted in terms of Computational Thinking and identity. The main defining feature of a case study is that the study is occurring in a bound system. This means the system can be described within parameters including location, time frame, and participants (Creswell & Creswell, 2018). This case study was considered exploratory. According to Yin (2003) there are three conditions to justify the use of an exploratory case study. First, a case study must answer a how or why questions. If you refer back to the research questions mentioned earlier, both questions are seeking to understand how knitting is impacting students understanding of Computational Thinking and their identity as a computer coder (Yin, 2003). Second, a case study is appropriate if the researcher has minimal control over the conditions of the environment the study is occurring in (Yin, 2003). During this study, while in a school setting, there were several outside occurrences that could not be controlled and impacted the environment of the study. These occurrences will be discussed in more detail later in this chapter but ranged from attendance issues to motivation. Third, a modern phenomenon in a social setting must be the focus of the study (Yin, 2003). While discussed in chapter one that Knitting Code itself is not a new occurrence, the idea that knitting could be used to teach computational skills is a new idea along with the growing need for more computer coders.

3.2.3 Purpose of the Comparison Group

One detail that is unique about this case study is a comparative group was used. This is referred to as theoretical replication (Yin, 2003, p. 47). The comparative group was necessary to answer the research questions. While not typical in a case study, a comparative group can be used in a multi-case study design if the researcher predicts the group will have contrasting results but for a predictable reason (Yin, 2003, p. 47). Since the comparative group was not receiving the same instruction, the researcher believed that results could differ based on instruction, especially regarding identity. Therefore, the two cases were the two groups: experimental and comparative.

3.2.4 Case Study Research Design

In a case study there are five components of research design including the guiding questions of the study, the propositions, the units of analysis, the logic of linking the resulting data to the propositions, and interpreting the findings (Yin, 2003, p. 21). The guiding questions of the study were mentioned earlier in this chapter and the propositions were discussed in chapter two. Specifically, the idea that knitting is similar enough to computer coding that it can teach similar skills. The unit of analysis are the groups (experimental and comparative). Due to the guiding questions of this study the “cases” will be the two groups: experimental and comparative. Also, since multiple cases are present this will be a multi-case designed case study. More on this will be discussed in the following “Participants” section. To link the data to the proposition descriptive coding was used in first cycle coding using a variety of types of data. This will be discussed in more detail later in this chapter. Finally, interpreting the findings was guided using pattern coding in second cycle coding. This will be discussed extensively in chapter four.

3.3 Setting

The school, that served as the setting of the study, was chosen for two reasons. First, the school was participating in the summer enrichment program which enabled the researcher to be able to carry out the investigation. Second, the researcher is also a teacher at this school, so, the researcher had access and clearance to work in the school. The school represents a middle school in north central Midwest United States. This school is the second largest school district in the state. The school district serves 41,476 students with 3,050 teachers in their 37 elementary schools, 12 middle schools, and 6 high schools (School Report Card Data, 2021). The student body is comprised of 46% of students that identify as White, 23.3% of students that identify as Black, 19% of students that identify as Latinx, and 4.8% of student that identify as Asian. The middle school the study occurred in is 1 of the 12 middle schools and serves 761 students between 6th and 8th grade. According to the school's overview 47% of students are considered economically disadvantaged and 53% of students are considered non-economically disadvantaged. Additionally, of the 761 students, 57.5% of students identify as White, 17.3% of students identify as Black, 12.6% of students identify as Latinx, and 12.6% of students identify as other (School Report Card Data, 2021).

3.3.1 Summer Enrichment Program Overview

The study, while conducted in middle school in north central Midwest United States, was conducted during a summer program, not during a traditional school year.

This was necessary due to the abstract approach to teaching that did not fit into the curriculum mandated by the district and state during the school year. After the 2020-2021 school year the school district decided to create a summer program to enable students a chance to either (a) receive credit recovery, (b) receive enrichment, or (c) reinforce ideas that students did not learn sufficiently through virtual learning. The summer enrichment program was comprised of two, three-week sessions. Each school had the ability to decide what they wanted their summer enrichment program to look like, and as a result a lot of flexibility was allowed in terms of teaching and student attendance.

The core subjects were taught by content teachers for students that needed credit recovery or reinforcement of concepts learned through the school year. These classes were grade specific due to the content being grade specific. Enrichment courses were offered for students that did not need credit recovery or did not need credit recovery in all four core subjects. These enrichment courses were determined by the teacher teaching them and that is where the Knitting Code program was conducted. These classes were a mix of 6th-8th graders. The teachers varied between the two sessions and even in the sessions, but during both sessions there were nine classes offered, taught by nine teachers. There was one additional teacher present that oversaw the program. All students that attended the summer enrichment program were previously districted for the school.

3.3.2 Summer Enrichment Program Schedule

Each school could determine their daily schedule. The middle school where the study was conducted determined that students would attend four classes a day with lunch between the third and fourth class. Each class was forty-eight minutes and the day lasted from nine am to one pm. A typical day in the summer enrichment program started with students being allowed in the building at 8:45am. As students arrived, they were given a free pre-packaged breakfast and went to their first class to eat their breakfast. Between 9:00am-9:05am teachers would start their instruction. At the end of forth-eight minutes students would transition to their next class with a three-minute break between classes to allow for transition and bathroom breaks. Students would then attend their second hour and third hour classes that would also last forty-eight minutes with a three-minute transition. At the end of third hour students would walk to the cafeteria to eat either their individually brought lunch or the free school provided lunch. The teachers would alternate days to monitor lunch. At the end of lunch students would go to their fourth-hour class that would last forty-eight minutes. Dismissal announcements would come on at 1:00 pm, and students would be dismissed.

3.4 Participants

The sixty-nine students that were assigned to the Knitting Code class were randomly assigned. As mentioned previously, students that attended the summer enrichment program were a mix of students that needed credit recovery, wanted summer enrichment, or needed reinforcement for content taught through the year. Students were assigned based on these needs and schedule allowance. Thus, the researcher had no impact on who was assigned her class. Additionally, some students she had taught previously and some she had not.

3.4.1 Demographics

Over the six total weeks the teacher taught sixty-nine students, but only twenty-five students returned completed ascent/consent forms. Of the sixty-nine students, 46%

identified as Black, 35% identified as White, 10% identified as Latinx, and 9% identified as Asian. Also, 56% identified as male and 44% of students identified as female. Of the twenty-five students that signed consent/assent forms 28% identified as Black, 28% identified as White, 24% identified as Asian, and 20% identified as Latinx. Of these students 60% of students identified as male and 40% identified as female. Additionally, of the twenty-five students three students had IEP's, five students had 504's, and one student was ELL.

The twenty-five students that returned assent/consent paperwork were placed in either an experimental or comparative group. The total number of students in the experimental group was thirteen. Of the thirteen students, 30.77% (n=4) identified as Black, 23.08% (n=3) identified as White, 30.77% (n=4) identified as Asian, and 50.38% (n=2) identified as Latinx. Additionally, 53.85% (n=7) identified as male and 46.15 (n=6) identified as female. There was also one student with an IEPs, two students with 504's, and one student was ELL. In the comparative group there were twelve students. Of the twelve students, 25% (n=3) identified as Black, 33.33% (n=4) identified as White, 16.67% (n=2) identified as Asian, and 25% (n=3) identified as Latinx. Additionally, 66.67% (n=8) identified as male and 33.33 (n=4) identified as female. There was also two students with IEPs and three students with 504's.

Student placement in the experimental or comparative groups was random based on which hour students were enrolled in their computer coding class. For the first session students in the first and second hour received the experimental instruction and students in third and fourth hour received the comparative instruction. During the second session students in the first and second hour received the comparative instruction and students in third and fourth hour received the experimental instruction. The teacher was not a part of the student assignments and the counselor assigning to classes did not know about the experimental or comparative groups used in this study.

There were twenty-five participants in this study and the participant were placed in either the experimental or comparative group. Each of these groups were considered a case, and as a result this study was a multi-case study design. These cases were the units of analysis mentioned earlier (Yin, 2003). The purpose of a multi-case design is to look at multiple cases that either (a) undergo similar tests and will receive similar results (literal replication), and/or (b) look at cases that will develop different results but are predicted to do so before the study begins (theoretical replication). During this study both components were used and that is why there was both an experimental and comparative group were used.

The experimental group was comprised of thirteen cases and students received computer coding instruction and knitting instruction. The comparative group was comprised of twelve cases and students received the same computer coding instruction, but instead of knitting instruction they received coding “unplugged” instruction. The two groups were necessary based on the research questions. A baseline was needed to compare the experimental group against the comparative group to make conclusions about how combining knitting to computer coding instruction impacts computation thinking skills and identity. In the next section curriculum development will be discussed in more detail.

3.5 Curriculum Development

Since a coding and knitting curriculum did not currently exist the researcher had to create a curriculum that combines aspects of both computer coding and knitting. Previous research indicates that Scratch is an engaging age-appropriate approach to teaching computer coding (Resnick et al., 2009). As a result, the guiding curriculum that was used when creating the "Knitting Code" curriculum was the "Creative Computing" curriculum by the Harvard Graduate School of Education (2019). The "Creative Computing" curriculum is a research-based curriculum that uses Scratch. This curriculum was originally released in 2011 and then again in 2013 as an online workshop before being released in its finished, research backed, 2019 version. After creating the plugged component, the "unplugged" component was designed. For the comparative group, "unplugged" lessons from code.org were used. For the experimental group, similar knitting correlations were identified and used. For example, using loops in knitting instruction to teach loops in coding.

Besides choosing resources for the curriculum a curriculum guide was also needed. The guide used for this study was a book titled *Step Into STEAM* by Bush and Cook (2019). The focus of this book is equitable STEAM education that combines meaningful problems, use of reform STEM techniques, and providing access for all students (Bush & Cook, 2019). The first component of using meaningful problems includes using inquiries that are open ended and do not have one correct solution. As a result, each student can inquire about the problem on a level that is comfortable for them. Problem-based learning will be used to achieve this first component (Bush & Cook, 2019). During problem-based learning students are presented with a problem and the need for students to learn the content is necessary to solve the problem (Bush & Cook, 2019). By using problem-based learning students learn to transfer and apply concepts learned to solve the original problem. The main problem that will be used for this curriculum is the student's favorite band has asked them to create a music video, but due to Covid 19 no one can meet in-person. Instead, the student needs to create an animated music video using Scratch. Problem-based learning has multiple levels that include inquiry reflection, and communication. This level of problem-based learning will be indicated on the curriculum.

Secondly, the use of reform STEM techniques will include meaningful discussion, posing purposeful questions, supporting students struggling, and engaging students in inquiry. Additionally, STEM will be the main framework with the art component added in to assist in learning. In the curriculum this will be attained by first listing the educational computer science standards taught. By listing the standards, the researcher/teacher will be able to keep the main STEM concept as the focus. The day's lesson and the incorporation of art will also be listed so that the instruction is meaningful. Additionally, to create meaningful discussion and pose purposeful questions the researcher/teacher will list potential questions that will be used and asked as a guide on the curriculum. Finally, to assist with engaging struggling or excelling students, intervention and enrichment opportunities will be listed on the curriculum.

The third component is providing access to all students. Bush and Cook (2019) argue that the STEAM program should be available to all students. This is accomplished by first making the program open to any student that wants to enroll. The summer enrichment program was offered to all students and did not have STEM or STEAM listed as a special emphasis. Additionally, any inquires that are required should have a low floor and high ceiling so that it is accessible to all (Bush & Cook, 2019). This is accomplished in the curriculum by not designating there is only one correct solution, but instead encouraging students to find a solution that works for them. Second, there is a differentiation section listed on the curriculum if differentiated instruction is needed.

Besides developing a curriculum and using a curriculum guide, the curriculum was traded back and forth with a content expert in Computer Science. This enabled the research the ensure that the curriculum was sound in terms of teaching computational skills and simple computer coding. Multiple adjustments were made during this time including connecting the content to computer science content standards and developing potential student questions. The curriculum is listed in the appendix.

3.6 Constraints in Study

As mentioned earlier a case study is appropriate if the researcher has minimal control over the conditions of the environment the study is occurring in (Yin, 2003). While the summer enrichment program occurred in a middle school, the researcher had minimal control over influences that impacted her study. There were several incidences that occurred outside of the researcher's control that could have impacted the study.

3.6.1 Impacts on Study

First, since there was no attendance requirement for participants in the summer enrichment program there was inconsistent attendance. This had the biggest impact on the study. Students would miss several days of instruction at a time and due to the short three-week time frame, there was minimal time for students to get caught up. Additionally, since the summer enrichment program occurred during the summer, students would disappear for a week to go on a family vacation. Thus, that student would miss a week of instruction and become behind. There would also be gaps in data collection for those individuals.

A third constraint was late busses/late arrivals. This was especially true for the first session. Students would arrive thirty to forty minutes into the first class which only lasts forty-eight minutes. This only allowed for ten to twenty minutes of instruction. This was a daily occurrence during the first session but did improve during the second session.

A fourth disruption to the study was schedule changes. Some students' schedules were switched mid-session due to developing needs or peer conflict. The issue with this is that three students were switched from an experimental instruction class to a comparative instruction class. While the students were able to resume instruction without issue, the researcher was not able to use all their data because they did not complete either curriculum completely.

Another constraint was unexpected disruptions. While the schedule was set, there were slight disruptions such as extending lunch a few minutes so students could eat outside and on the last day allowing students to stay outside and eat popsicles during the last hour. While there were not specific incidents that caused issues, the combination of these small disruptions impacted data collection.

Another constraint, while known about ahead of time, was time constraints. The researcher was limited by a three-week session (fourteen days total for each session) and fitting in all components of the study was a challenge. While content was cut down beforehand and a curriculum was used to guide the study, it was still unknown how much time would be needed. More time could have been used on any given day for more practice, journaling, or discussion.

The final disruption to data collection was students' motivation. Several students were not happy they were going to school during summer. This was repeated verbally to the teacher-researcher on several occurrences. Since there was no requirement that students had to complete the work, unless they were there for credit recovery, the teacher had to rely largely on engagement and relationships to get students to participate. Overall, participation was high, but students' lack of motivation emerged sporadically through the study, and as result some students did not put in their full effort in the class.

3.6.2 Teacher as the Researcher

Another constraint to the study was that the teacher was also the researcher. The purpose of this decision is due to the set of skills the researcher possesses. The researcher is a certified teacher and understands the pedagogy of the coding program, but she also has vast experience in knitting and other similar crafts. Since the researcher will also be the teacher several cautionary steps will be implemented to reduce bias. First, when the teacher/researcher attains consent/assent she left the room and had another teacher (that was approved by the IRB) collect this information. The assents/consents were kept safe and given to the teacher/researcher at the conclusion of the study. This reduced any feelings that someone must participate to attain a grade, and the teacher did not have bias if a student was not choosing to participate. Second, triangulation when collecting data to reduce bias. Finally, when data was coded the researcher gave each case a code and scrambled the cases, so the researcher did not know which group the case belonged to. This reduced the bias of the researcher coding the data a certain way to make a case stronger in support of an emerging theme.

3.7 Sources of Data

In a case study multiple sources of information need to be collected through multiple data collection sources (Creswell & Poth, 2018; Yin, 2003). Common data collection sources include documents, interviews, observations, questionnaires, artifacts, and audiovisual material. For this study all five of the six sources were used. Interviews were omitted due to constraints that the researcher is also the teacher. Instead, students responded to a prompt and spoke to a camera in an audiovisual interview. The survey was given at the beginning and end of the study, but other than that the four remaining data collection sources were used daily or every other day.

3.7.1 Documents/Student Journals

The documents for this study included a journal that students used daily during instruction to draw out designs and answer prompts. Questions for this journal are designated in the curriculum in the section listed as assessment. The journals were placed in a three-prong folder and given out daily. At the beginning of each class the teacher gave out the journal papers for the day to be added to the back of the folder. The journals were then collected daily for storage purposes and at the conclusion of the program for analysis. The instructions for the daily activities and journaling prompts were taken from the “Creative Computing” curriculum by the Harvard Graduate School of Education

(2019). Both the experimental and comparative group were given the same journal papers and answered the same prompts since both groups completed the same computer coding activities. Examples of the journal papers can be found at this website (https://creativecomputing.gse.harvard.edu/guide/curriculum.html) as part of the "Creative Computing" curriculum by the Harvard Graduate School of Education (2019).

3.7.2 Observations

Observations were made daily and recorded using a semi-structured observation form. This form allowed the researcher some freedom when making observations but was also semi-guided so that the researcher collected the necessary data. A new form was used for each class, each day. So, four observations forms were used daily. Once the forms were completed, they were stored in a three-ring binder in order of time collected. The form was modified from examples and guidelines presented in "Qualitative Inquiry and Research Design" by Creswell and Poth (2018). Since the form was semi-structured the same form was used for both the experimental and comparative classes. An example of a blank observation form can be found in the appendix.

3.7.3 Identity/Attitude Survey

A computer science identity and attitude survey was used to determine any changes in identity towards computer science during the course of the program. The identity survey (SCAIS Survey) had been validated using principal component analysis (Washington, Grays, & Dasmohapatra, 2016). The survey was created due to a rising need to better understand confidence, interest, gender, professional, and identity of those involved in computer science (Washington, Grays, & Dasmohapatra, 2016). Additionally, at the end of the survey demographics were collected. The identity survey was given at the beginning and end of the program. The survey was adapted from the CSAIS survey and was not changed except for formatting and an addition of a demographics section. Both groups received the survey at the beginning and end of a three-week session. The one exception is that during the last hour of the first session students did not take the survey due to an end of session celebration that was decided last minute. An example of the CSAIS survey can be found in the appendix.

3.7.4 Audiovisual Interviews with Bebras Challenges

Since interviews were not possible due to constraints, audiovisual interviews were used by students recording themselves responding to a prompt using Flipgrid. These interviews focused on students using Computational Thinking skills. To guide the audiovisual interviews Bebras Computational Challenges were used. Audiovisual interviews were chosen in addition to the journals to alleviate any differences in reading and writing abilities between students.

Every other day students were given a Bebras Computational question and responded by recording their answer. These questions were taken from past Bebras Computation Challenges and focus on computation skills. The Bebras Computation Challenge is an Australia Based international competition that targets students between the ages of six to eighteen. If a student chooses to participate, they are given fifteen multiple choice questions that require computational skills to solve and forty-five minutes to complete (Bebras, 2021). For the audiovisual interviews students were first presented with the Bebras Computational Challenge question, asked to solve, state their solution, and then explain their reasoning using Flipgrid. Additionally, students were asked explain what they learned and how that "unplugged" activity applies to what was learned. These questions were pre-designated in the curriculum.

The audiovisual prompts had two questions. The first question asked students to explain their reasoning for picking the answer they did in their Bebras Computation Challenge. The Second question asked students to explain what they had learned during the lesson. Both the experimental and comparative groups received the same prompts because the lessons focused on learning the same computational skills. The differences were the experimental sometimes referenced knitting skills and the comparative group did not. Examples of the Bebras Challenges used during the study can be found in appendix.

3.7.5 Artifacts

Artifacts were developed in the form of Scratch projects, coding "unplugged" activities, and knitted items. The artifacts created were developed naturally as a requirement of the curriculum. The researcher was able to set up a class on Scratch where students uploaded all their Scratch projects. Additionally, pictures were taken during coding "unplugged" activities and any sketches completed were placed in student journals. Finally, while students chose to take home their knitting projects, the researcher took pictures of knitted items in progress. Students were not able to complete their final kitted project due to time constraints but were able to get started.

Knitting project ideas were taken from the Ravelry website (Ravelry.com). The Scratch project ideas were based on the researcher developed curriculum that was created from the "Creative Computing" curriculum by the Harvard Graduate School of Education (2019). Both the experimental and comparative group completed the same Scratch projects and thus had the same Scratch artifacts. The difference in the artifacts is that the experimental group created knitted artifacts and the comparative group generated coding "unplugged" artifacts.

3.8 Data Collection

As mentioned earlier, each session was three weeks, with the first week only being four days, and there were two sessions. The experimental group received coding instruction and knitting instruction while the comparative group received the same coding instruction, but instead of knitting they received coding "unplugged" instruction. Before the study started several steps had to occur. First, the researcher had to get approval from the school district and the IRB committee to conduct her study. This required going back and forth over details about the study. The second step the researcher completed was gathering materials and setting up the student journal folders. Finally, before the study started the researcher read through her literature and made a set of precoding terms. In the below table (Table 3.1) what data was collected each day can be seen. For a more detailed list, please reference the curriculum in the appendix.

3.9 Data Analysis

Data analysis served as a way for the researcher to use the data to answer the research questions. Since this study was an explanatory case study the focus was on answering the how and why of the research questions. Determining internal validity was necessary in establishing a quality case study and these steps occurred during the data analysis phase (Yin, 2003). Internal validity was established by pattern-matching, explanation building, addressing rival explanations, and using logic models. Besides internal validity, high quality case study analysis included analyzing all data, addressing all major rival explanations, answer the pre-established research questions, and the researcher used his/her expert knowledge on the topic (Yin, 2003, p 137).

Qualitative data was coded to determine if themes would emerge. Before the themes could be synthesized the data had to undergo primary and secondary coding. During primary coding the data was analyzed and during secondary coding the data was synthesized (Saldana, 2021). Codes were developed from the data, then categories were developed from the codes, and themes were developed from the categories. At the conclusion of the data analysis a theory was developed

Table 3.1

Daily Data Collection Methods

Day and Purpose	Experimental	Comparative
Day 1 (Lesson 0) Introduction to coding – Students set up their Scratch account and were introduced the goal of the class.	- Consent/Assent form - CSAIS Survey	- Same as experimental
Day 2 (Lesson 1A) Introduction to Scratch - Students completed the Scratch tutorial and worked on the ten-block challenge	- Daily Journal Sheets - Ten-Block project on Scratch - Daily observations	- Same as experimental
Day 3 (Lesson 1B) Introduction – Taught sequence through cast-on directions (experimental) or writing dance directions (comparative)	- Daily observations - Knitting cast-on pictures - Bebras/Audiovisual journal	- Daily observations - Tik Tok dance directions - Bebras/Audiovisual journal
Day 4 (Lesson 2A) Debugging Code – Debugging taught through Scratch Simulations	- Daily journal sheets - Daily observations	- Same as experimental
Day 5 (Lesson 2B) Debugging – Reviewed debugging and sequence through learning the knit stitch (experimental) or completing the Picture This activity (comparative)	- Daily observations - Knit stitch pictures - Bebras/Audiovisual journal	- Daily observations - Picture This activity drawings/directions - Bebras/Audiovisual journal
Day 6 (Lesson 3A) About Me – Students created a collage on Scratch about their interests	- Daily journal sheets - Daily observations - About Me Scratch project example -	- Same as experimental
Day 7 (Lesson 3B) Explore – Students will explore Ravelry projects (experimental) or Scratch projects (comparative) and have a discussion reviewing concepts learned	- Daily observations - Bebras/Audiovisual journal	- Daily observations - Bebras/Audiovisual journal

Day and Purpose	Experimental	Comparative
Day 8 (Lesson 4A) Build-a-Band – Students will create a band on Scratch that uses different instruments to show parallelism	- Daily Journal Sheets - Daily observations - Build-a-Band project on Scratch	- Same as experimental
Day 9 (Lesson 4B) Parallelism – Students will wither practice knitting and have a conversation about the different steps occurring (experimental) or "code" a friend to complete different tasks at the same time (comparative)	- Daily observations - Bebras/Audiovisual journal - Pictures of knitting practice	- Daily observations - Bebras/Audiovisual journal - Pictures of "code" a friend activity
Day 10 (Lesson 5A) Loops – Students will complete the Its Alive project on Scratch to show movements using loops	- Daily Journal Sheets - Daily observations - Its Alive project on Scratch	- Same as experimental
Day 11 (Lesson 5B) Loops Review – Students will look at knitting patterns, choose an end project, and start working (experimental) or create an obstacle course and use loops to code their friend to complete (comparative)	- Daily observations - Bebras/Audiovisual journal	- Daily observations - Bebras/Audiovisual journal - Pictures of obstacle course activity and written directions by students
Day 12 (Lesson 6A) Music Video – Students will combine their knowledge/skills to create a music video on Scratch	- Daily Journal Sheets - Daily observations - Music Video project on Scratch	- Same as experimental
Day 13 (Lesson 6B) Music Video – Students were either given continued time to work on their knitting project (experimental) or music video project (experimental and comparative)	- Daily Journal Sheets - Daily observations - Music Video project on Scratch - Pictures of knitting project in process (no completed pictures available)	- Daily Journal Sheets - Daily observations - Music Video project on Scratch
Day 14 (End) Conclusion – Students completed a gallery walk using the music video Scratch projects. Students provided feedback. Afterwards, students completed the post SCAIS survey	- Daily Journal Sheets - Daily observations - Music Video Scratch project - Post- CSAIS Survey	- Same as experimental

Data coding occurred at all stages of the research. Before the research was conducted pre-coding occurred. During pre-coding, items of interest from articles, that were perceived as future codes, were highlighted, or marked in some way to be accessible for future use (Saldana, 2021). Next, data coding began when the research began with preliminary coding. During preliminary coding words or phrases that started to emerge during the study were noted (Saldana, 2021). There was also a section on the daily observation form to note preliminary codes. By completing pre-coding and preliminary coding a data base of possible codes were developed before the analysis and synthesis phase.

Upon completion of the research, primary and secondary coding occurred. While a set number of codes were not be specified, an effort was made to not create so many data codes the data became overwhelming. Additionally, a free CAQDAS program called QDA Miner Lite was used to manage the data and data code collection. During primary coding several coding methods were used including descriptive coding and in vivo coding (Saldana, 2021, p. 97). Upon the completion of primary coding, but before moving to secondary coding, the codes developed were put in a word landscape program. A word cloud and graph were used to visually see repetitive codes that emerged. Finally, secondary coding occurred using pattern coding (Saldana, 2021, p. 97). The data underwent primary and secondary coding many times until clear themes emerged. Additionally, survey data was analyzed using the Wilcoxon Signed Rank Test because the surveys served a tool to show a change over time using matched samples from the pre and post survey.

3.9.1 Precoding

During precoding the researcher used deductive coding techniques to generate a list of potential code words before the study started. This was considered appropriate because the research, literature, and research questions lent themselves to certain code (Saldana, 2021, p. 40). Literature used in the literature review was used to develop a list of codes. As the researcher came across the code more frequently, she would increase the size of font. At the end of coding all the literature she used the “biggest” words as her starting codes. Additionally, the researcher lumped codes that were similar such as sequence and algorithm or mistake and debug. The end goal of the pre-coding was to develop a list of codes, but also to keep that list manageable and not overwhelming. Pre-codes include:

- coding is new literacy
- “unplugged”
- transfer knowledge
- creation of knowledge
- play
- engaging learning
- connect with lives/real life
- foreign language/
- coders vs programmers
- similar peers
- stereotypes
- marginalized
- stereotypic view
- stereotypic artifacts
- identity
- interactive second language
- critical thinking
- processing and learning time
- what to learn
- how to learn
- algorithm
- context

3.9.2 First Cycle Data Coding

After data was collected the first cycle coding process started. Two types of first cycle coding were used. First, descriptive coding was used since data included observations, student journals/documents, and artifacts. Additionally, In Vivo coding was used because of the use of audiovisual interviews.

The first step for the first cycle coding was to transfer the data into QDA miner Lite to me analyzed. First transcripts were typed out for the audiovisual interviews. Next, pictures and screen shots were added to the audiovisual interview transcripts. After quotes from student journals were added to the bottom of the transcripts. Additionally, the semi-structured observation forms were turned into a word document. Once all the data was digitized, the documents were uploaded to QDA Miner Lite.

Before starting the data coding process, the researcher gave the different students a case number and then scrambled the students so the researcher would not know if she was coding a student from the experimental or comparative group. This was done to remove any bias the researcher may have had. Once ready, the researcher read through the different cases of data and applied the preset codes to the data. Once done, the researcher repeated the process two more times to make sure everything had been coded. The below results show a word cloud (Figure 3.1 and Figure 3.2) and code frequencies (Figure 3.3 and Figure 3.4) of the codes that emerged.

Figure 3.1

Word Cloud of Experimental Coded Data

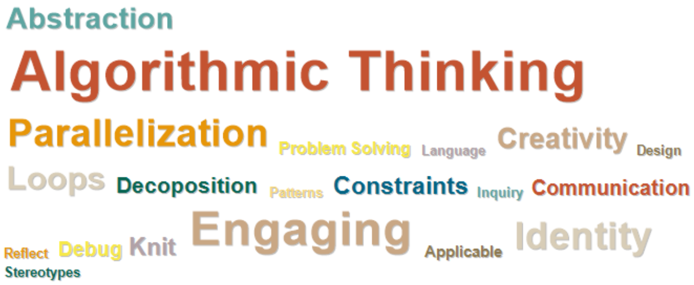


Figure 3.2

Frequency of Code for Experimental Group Coded Data

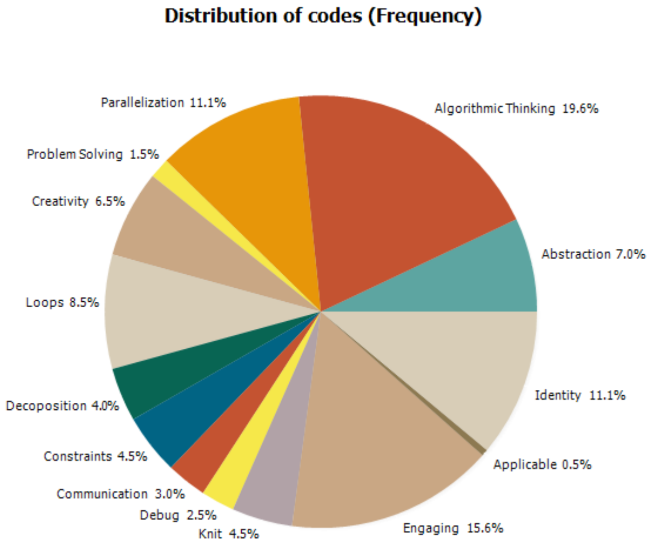


Figure 3.3

Word Cloud of Comparative Coded Data

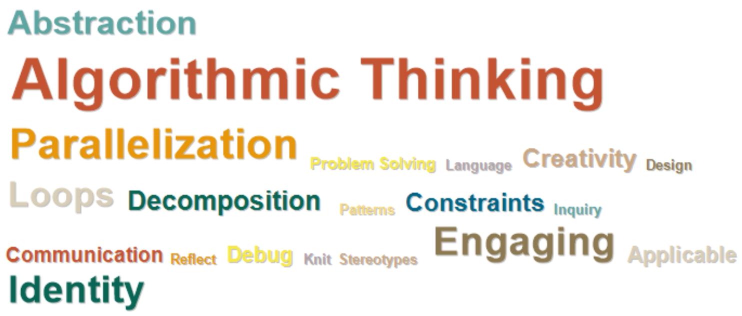
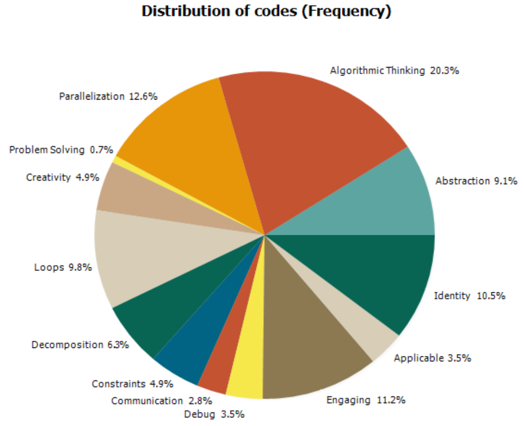


Figure 3.4

Frequency of Code for Comparative Group Coded Data



Results showed that Algorithmic Thinking was highest in both groups of students. The second highest frequency code for the experimental group was engagement while it was parallelization for the experimental group. For the third highest frequency for the experimental group was parallelization and for the comparative group it was engagement. Then both the experimental and comparative group had the same fourth most frequently used code which was identity. Overall, for both groups the top four frequent codes were

Algorithmic Thinking, parallelization, engagement, and identity. This information was used for the second cycle coding.

3.9.3 Second Cycle Data Coding

Pattern coding was used for second cycle coding. The purpose of pattern coding as a second cycle coding method was to take the various codes and to develop an explanation or theme (Saldana, 2021, p. 322). Pattern coding examines the cause-effect relationships found in data. Since this case study is an explanatory study and is also examining cause-effect relationships, pattern coding was deemed appropriate.

Before themes could be developed the data was categorized based on patterns that developed. The main pattern that was established was that in both the experimental and comparative group the same four major code frequencies emerged. Those four codes (Algorithmic Thinking, engagement, parallelization, identity) became the main categories. Then the researcher looked at the remaining codes and categorized them into one of the four major codes based on themes of what students were trying to accomplish when the code was assigned.

Table 3.2 Table shows the codes that developed each category and an explanation of what the category means. (Note: italicized codes represent new codes that developed after pre-coding occurred)

Table 3.2

Categories from Coded Data

Codes	Categories	Definition/Explanation
Algorithmic Thinking, Abstraction, *Problem Solving*, Decomposition, Constraints, Debug	Algorithmic Thinking	Category includes components of making a sequential working computer code
Engaging, Applicable, *Creativity*	Engagement	Category includes components that keep students invested in what they are doing
Parallelization, *Design*, *Loops*	Parallelization	Category includes codes that enables the computer code to occur in working order with another computer code so multiple events can occur at the same time
Identity, Stereotypes, Communication	Identity	Category includes codes that shows student taking ownership for their projects.

Once the data codes were turned into categories they were then turned into themes. The researcher examined the four categories and realized two themes emerged. Those themes were related to the research questions. The first theme that emerged included the Algorithmic Thinking category and the parallelization category. These are both type of Computational Thinking skills and thus the first theme that emerged was that students were demonstration Computational Thinking skills in both the experimental and comparative group.

The second theme that emerged included the engagement and identity categories. Both categories dealt with how the participants viewed themselves as able and willing to computer code and work with computers.

Table 3.3 Table shows the categories that developed each theme and an explanation of what the category means. (Note: italicized codes represent new codes that developed after pre-coding occurred)

Table 3.3

Themes from Coded Data

Categories	**Themes**	**Explanation**
Algorithmic Thinking Parallelization	Use of Computational Thinking Skills	Since Algorithmic Thinking and parallelization are both Computational Thinking skills students were using Computational Thinking skills in their daily lessons during their instruction.
Engagement Identity	"I can be a computer scientist"	Categories dealt with how the participants viewed themselves as able and willing to computer code and work with computers.

3.10 Quality Case Study Design

To develop a quality case study four tests were used. These tests include (1) construct validity (2) internal validity (3) external validity and (4) reliability (Yin, 2003, p. 33).

3.10.1 Construct Validity

The first test was construct validity and this test ensures that the data that was being collected is applicable to the questions being asked. To ensure this is happening multiple sources of evidence and a chain or evidence must be established during the data collection phase. Additionally, construct validity can be developed by having key informants review the case study report during the composition phase (Yin, 2003).

To ensure construct validity is occurring the researcher developed a study that did collect multiple sources of data and that data was organized, handled, and stored in an organized method that received prior IRB approval. Additionally. The case study has been reviewed by key informants including the researcher's advisor and PhD committee to ensure the validity.

3.10.2 Internal Validity

Internal validity was necessary to determine in an unknown variable impacted results compared to the known variable. Due to the nature of a case study this can be a common problem due to the researcher being unable to control the environment. To ensure internal validity several tactics were used such as pattern-matching, explanation building, addressing rival explanations, and using logic models. All of these occurred during the data analysis phase (Yin, 2003, p34).

Pattern matching occurred when the researcher compared results with hypothesized results. In other words, the patterns that emerge matched the predicted patterns. (Yin, 2003). Explanation building, while a type of pattern matching, was when the researcher used the data collected and the patterns developed to make a causal statement regarding the phenomena (Yin, 2003). This method is often considered more in depth than pattern matching in general, so both are recommended.

Addressing rival explanations is when the researcher is aware of other explanations and addresses these other explanations during the analysis phase (Yin, 2003). Logic Models are when cause-effect relationships are studied over a period. Due to the constraints of this study the period of study was three weeks. The purpose of logic models was to observe events and compare to predicted events (Yin, 2003). This could also be considered a form of pattern matching, but since the observations occur over time instead of just at the end of the study the method differs slightly.

3.10.3 External Validity

External validity determines if a case studies' results can be generalized beyond the cases that is being studied (Yin, 2003, p. 37). In other words, if a case studies' results can be beneficial to promote future studies. To promote external validity a multi-case study design was beneficial. By using multiple cases, the researcher was able to make a more generalizable statement since the same trends were seen in multiple cases. Thus, external validity is established during the research design phase. For this research a multi-case study design was used with each of the two group, experimental and comparative, being a case. Determining trends and themes among the different cases were a focus to establish external validity.

3.10.4 Reliability

A reliability tests should enable another researcher to read the research design and replicate the research design (Yin, 2003). Due to the nature of this case study and the development of a curriculum it would be possible for another researcher to duplicate this study using the resources the researcher created. The other researcher then could follow a similar protocol when collecting evidence and thus ensure reliability. While all the information is available a reliability study was not conducted.

3.10.5 Reduce Bias

Additionally, to reduce bias several steps were taken. First, when the teacher/researcher attained consent/assent she left the room and had another teacher (that was approved by the IRB) collect this information. The assents/consents were kept safe and given to the teacher/researcher at the conclusion of the study. This reduced any feelings that someone must participate to attain a grade, and the teacher did not have bias if a student was not choosing to participate. Second, triangulation when collecting data to reduce bias. Finally, when data was coded the researcher gave each case a code and scrambled the cases, so the researcher did not know which group the case belonged to. This reduced the bias of the researcher coding the data a certain way to make a case stronger in support of an emerging theme.

3.11 Summary

The case study occurred during two, three-week, sessions where each case received a total of fourteen, forty-eight-minute lessons. Additionally, the study occurred in a public middle school during a summer enrichment program. The researcher taught four classes each session. Two of the classes received computer coding instruction and knitting instruction. This was the experimental group. The other two classes received computer coding instruction and coding "unplugged" instruction. This was the comparative group. The teacher-researcher designed the curriculum based on the "Creative Computing" curriculum by the Harvard Graduate School of Education (2019), her own knowledge of knitting, and with the help of a content expert in computer science.

The data for this case study came from a variety of sources and was analyzed using qualitative data coding. Data included student journals, class observations, identity surveys, audiovisual interviews based on Bebras Challenges, and student artifacts. Multiple sources of data were used to build validity in the case study design and to discourage bias (Yin, 2003). Before the study started precoding was conducted using deductive methods to develop a list of potential code words. During first cycle coding the data was analyzed using descriptive and In Vivo coding (Saldana, 2021). From that data four categories emerged that were named (1) Algorithmic Thinking (2) Engagement (3) Parallelization (4) Identity. During second cycle coding pattern coding was used. As a result of pattern coding two themes emerged that were called (1) Use of Computational Thinking Skills (2) "I can be a computer scientist".

Chapter 4 FINDINGS

This case study was guided by two research questions based off the emerging idea that knitting and coding are similar. Due to this similarity, statements have been made that knitting can promote computer coding skills (Buckner, 2015; Roberts, 2019). This study was set up as an exploratory case study design due to its focus on answering "how" and "why" questions in a hard to control scenario using a modern phenomenon (Yin, 2003, p.1). Overall, this case study wanted to increase understanding between the relationship between knitting and computational skills (Yin, 2003). Due to this focus a multi-case design was needed that also included a comparative group used for theoretical replication. Additionally, while the total number of participants was smaller than other forms of experimental study design, the data is still valid because case studies are focused more on how cases are impacted over time and less concerned with frequencies or incidents (Yin, 2003).

The case study occurred during two, three-week, sessions where each case received a total off fourteen, forty-eight-minute lessons. Additionally, the study occurred in a public middle school during a summer enrichment program. The researcher taught four classes each session. Two of the classes received computer coding instruction and knitting instruction. This was the experimental group. The other two classes received computer coding instruction and coding "unplugged" instruction. This was the comparative group. The teacher-researcher designed the curriculum based on the

“Creative Computing” curriculum by the Harvard Graduate School of Education (2019), her own knowledge of knitting, and with the help of a content expert in computer science.

The data for this case study came from a variety of sources and was analyzed using qualitative data coding. Data included student journals, class observations, identity surveys, audiovisual interviews based on Bebras Challenges, and student artifacts. Multiple sources of data were used to build validity in the case study design and to discourage bias (Yin, 2003). Before the study started precoding was conducted using deductive methods to develop a list of potential code words. During first cycle coding the data was analyzed using descriptive and In Vivo coding (Saldana, 2021). From that data four categories emerged that were named (1) Algorithmic Thinking (2) Engagement (3) Parallelization (4) Identity. During second cycle coding pattern coding was used. As a result of pattern coding two themes emerged that were called (1) Use of Computational Thinking Skills (2) “I can be a computer scientist”. The end goal of a case study is to generalize what was discovered, not particularize (Yin, 2003, p11). As a result, some trends were discovered regarding the original research questions.

4.1 Research Question One: Computational Thinking Skills

During the literature review phase of the study, the literature revealed that Computational Thinking skills were often mentioned as needed skills for those interested in pursuing computer science. Computational Thinking was defined as a cognitive process where an individual takes a big problem and divides the problem into smaller sections that can be solved through a set of steps (Lee & Junoh, 2019; Ricketts, 2018; Sung, 2019; Tonbuloglu & Tonbuloglu, 2019). Also, as stated before, key elements of

Computational Thinking include (1) abstraction and automation (2) systematic process and information (3) symbol systems and representations (4) algorithmic notions of flow control (5) structured problem decomposition (6) iterative, recursive, and parallel thinking (7) conditional logic (8) Efficiency and performance constraints (9) Debugging and systematic error detection (Grover & Pea, 2013).

4.1.1 Data Analysis

Computational Thinking has become popular, particularly in computer science education. Sung (2019) states that Computational Thinking has a close relationship with technology and engineering education because similar processes occur. "The use of technology involves Computational Thinking skills, and computer science is used for the acquisitions of Computational Thinking skills" (Tonbuloglu & Tonbuloglu, 2019, p. 404). The importance of computational skills in computer science and the idea that knitting can teach computer coding lead to the development of the first research question of the study which is…

How does learning to knit while learning to code facilitate the understanding of Computational Thinking skills including abstraction, Algorithmic Thinking, and understanding of parallelization in adolescent students?

Overall, the data suggest that Computational Thinking skills were taught through knitting. While there is no evidence to support the approach that using knitting is better or worse, the purpose of the study was just to determine if there would be similar success. Since there was similar success, the researcher can determine that knitting can be used as an instructional method to teach Computational Skills in computer science. During the pre-coding phase several of the Computational Thinking skills were identified as potential codes due to their frequency of use in literature. These terms appeared not just in articles referring to computation thinking skills, but also in articles referring to computer science, computer coding, and problem solving.

Terms related to computation thinking skills that were pre-coded included Algorithmic Thinking, Abstraction, Decomposition, Constraints, Debug, Parallelization. Additionally, design, loops, and problem solving, which are related to computation thinking skills, were added as during first cycle coding. Once the first cycle coding process started these data codes were categorized into two main categories named Algorithmic Thinking and Parallelization. Algorithmic Thinking and Parallelization were chosen as the main category names over the other Computational Thinking skills due to their frequency as codes in the data. The remaining codes that were related to Computational Thinking skills were then examined and placed into one of the two categories based on their purpose when coded in the data.

Table 4.1

Categories Developed from Codes

Codes	**Categories**	**Definition/Explanation**
Algorithmic Thinking, Abstraction, *Problem Solving*, Decomposition, Constraints, Debug	Algorithmic Thinking	Category includes components of making a sequential working computer code
Parallelization, *Design, Loops*	Parallelization	Category includes codes that enables the computer code to occur in working order with another computer code so multiple events can occur at the same time

After developing these two categories they were combined into one theme called the "Use of Computational Thinking Skills". The emergence of this theme shows that students were using Computational Thinking skills during their project creation.

Table 4.2

Themes Developed from Categories

Categories	Themes	Explanation
Algorithmic Thinking Parallelization	Use of Computational Thinking Skills	Since Algorithmic Thinking and parallelization are both Computational Thinking skills students were using Computational Thinking skills in their daily lessons during their instruction.

When comparing experimental group results against the comparative group, the experimental group received similar frequencies of coded data. This explains why different categories and themes were not developed for each group. This also suggests that the experimental group performed at a similar level as the comparative groups. The frequencies of use of coded data can be seen in figure 4.1 for the experimental group and Figure 4.2 for the comparative group.

Figure 4.1

Experimental Group Frequency of Codes

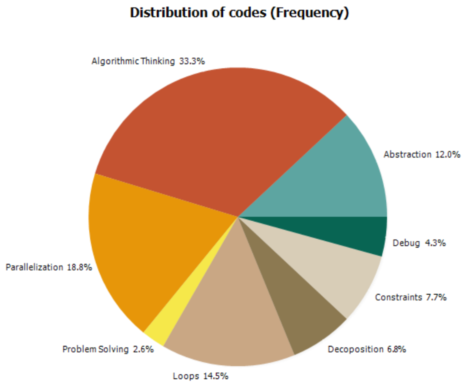
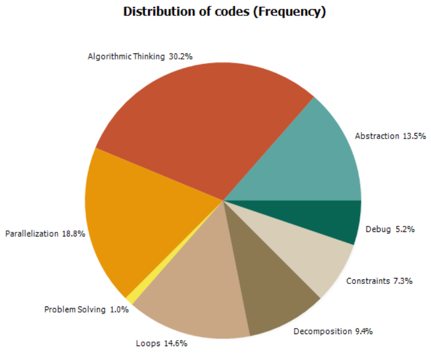


Figure 4.2

Comparative Group Frequency of Codes



To get a more in-depth understanding of Computational Thinking skills students developed an examination of student work is needed.

4.1.2 Student Examples of Algorithmic Thinking Category

The Algorithmic Thinking category contained several codes including Algorithmic Thinking, abstraction, problem solving, decomposition, constraints, and debug. Overall, when the codes are combined components of making a sequential working computer code emerge. To see how students developed Algorithmic Thinking a closer examination of the encompassing codes needs to occur.

Algorithmic Thinking is essentially a list of step-by-step procedures used to complete a task (Lee & Junoh, 2019). One example of Algorithmic Thinking can be seen in the “Dress Code” Bebras Challenge. An example of this Bebras Challenge can be found in the appendix. During the challenge students were given a branching decision flow chart. Students had to follow that flow chart and use step-by-step decisions to solve the problem of which beaver was not dressed correctly. Students were using Algorithmic Thinking if they were able to follow each step of the problem and make a decision to lead them to the correct answer.

In an audiovisual interview case E7, from the experimental group, was able to answer correctly using Algorithmic Thinking. E7 stated Beaver B was the beaver not dressed correctly. They stated their reasoning for this decision as, “I chose my answer because A has glasses and C has a yellow hat and D has a blue hat. The question asked which Beaver does not like the dress code, and B broke all the rules, so I think he doesn't like the dress code. To find the answer we had to look at every single part to find out which Beaver did like the dress code.” Similarly in the comparative group, case C7 was able to provide similar reasoning for selecting Beaver B stating that, “I chose Beaver B because they have neither classes or blue hat or a yellow hat.” In both the experimental and comparative cases the students were able to use similar Algorithmic Thinking to achieve the same answer.

Abstraction is another term makes up the Algorithmic Thinking category. Abstraction is defined as ignoring specific details to focus on the general idea (Sung, 2019). A more general idea of abstraction is that to drive a car (general idea) the user does not need to understand how the engine works (specific details). Several discussions about abstraction occurred in class and were recorded on the daily observation sheets.

In a class discussion, with the experimental group, the researcher asked students questions regarding how a sweater was made in relation to abstraction.

Researcher: What is needed to wear a knitted scarf?

Case E1: You need to know how to cast-on and knit.

Researcher: Anything else?

Case E4: You need to know how to count.

Researcher: Do you need to know how yarn is made to make a scarf?

Case E1: No

Researcher: Okay, does a person need to know how to knit a scarf to wear one?

Case E1: No…

Researcher: So, if we were removing un-needed information, what is not necessary for a person to know to wear a knitted scarf?

Case E2: A person doesn't need to know how the scarf is made to wear the scarf.

Case E1: You don't need to know how to knit.

Researcher: This is known as abstraction. Can you think of an example in Scratch?

Case E4: To play one of the pre-made songs on Scratch you don't need to know which notes are used, just how to place the sound block and choose the song.

Similarly, the researcher also asked the Comparative group some questions regarding abstraction.

Researcher: What games did you enjoy playing on Scratch?

C7: I liked the Flappy Bird game

C6: I liked the maze

C5: I also like the Flappy Bird game

Researcher: Okay, how is the Flappy Bird game made?

C7: I can look inside the game and find out.

Researcher: That is not needed unless you want to know. Did not knowing how any of the games you played impact your ability to play the game?

C5: No, because we just had to play the game, not make the game.

Researcher: This is known as abstraction when you can focus on the main idea and ignore the details. Can you think of an idea in Scratch?

C7: Could Scratch be abstraction because you don't need to know a coding language to create computer created programs?

In both the experimental and comparative groups students were able to identify that you could perform an action and not know all the details that make up that action. While this is a simplified explanation of abstraction both groups of students were able to take the concept and apply it to Scratch.

Problem solving, while a code used in the category of Computational Thinking Skills, was the least used. While students were able to problem solve during the sessions, the process was not specifically noted as much. This might be due because problem solving is a more overarching idea and involves other components such as debugging and decomposition. When coding the data the researched used the more specific terms and thus, problem solving was not used a frequently. Examples of problem solving that were noted were when students wanted their character to perform a certain action so the student had to think about which block to use. For example, in both the experimental and comparative group students struggled to get their first program to run. They were able to problem solve that they forgot to insert a signal to start box at the beginning of the code.

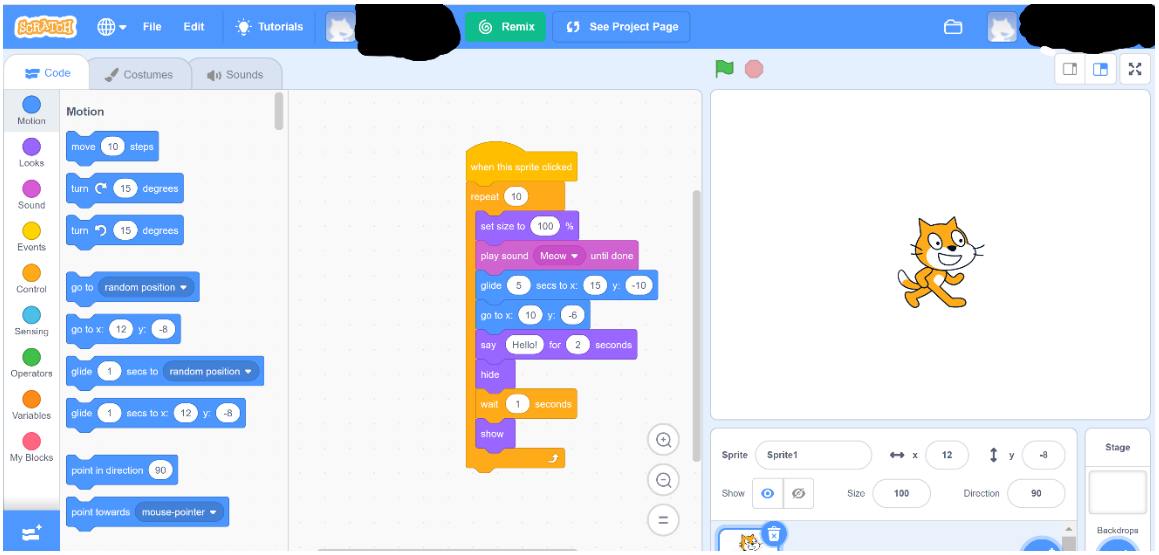
Another component of the Algorithmic Thinking category was decomposition. Decomposition can be defined as breaking problems down into smaller sections through the process of a flow chart. This process illustrates the logic from the start to the end when solving a problem (Sung, 2019). An example can be seen in the “Animation” Bebras Computational Challenge. An example of this challenge can be found in the appendix. During this challenge students were given an end image and asked to put the remaining images in order to produce the end image while only changing one component in each picture. This challenge is an example of decomposition because students had to break down components of the end picture to determine what differs between each picture. In other words, there was an end image and students had to look at one change that occurred from picture to picture to achieve the correct order of pictures.

The correct order of the pictures is BDCAE or answer four. Case E8, from the experimental group, stated that, “I chose answer four. The order is BDCAE. I chose 4 because I think that it is as smooth as it could be because the picture changed to wiggly to straight and then happy.” Additionally, case C10, from the comparative group, stated that, “For my answer I got #4 which is BDCAE. I did B first because it had the biggest ears and the biggest whiskers. It also had, like, the whiskers that are, like, squiggly and the rest of them didn't have that. I knew that D would go next because it still had those squiggly whiskers, but the ears were smaller, and then after D it was C because the noses are still the same. They are circular and then A because the noses are bigger, and the last one would have to be E.” In both examples, both the experimental and comparative group, were able to observe individual features and break down the problem of which order the pictures would occur in to make a smooth animation.

Constraints was another code that was included in the Algorithmic Thinking category. Constraints are similar to restrictions and can be observed in Lesson One in the “Ten Block Problem” Scratch challenge. Students were given ten blocks on Scratch and told they could only use the given blocks and they must use all blocks at least once. Student struggled with this challenge because they had just been introduced to Scratch and were still learning how to use the different blocks. Students also wanted to experiment with all the blocks, not just the ten listed. In Figure 4.3 an example of the challenge can be seen that was completed by E1 in the experimental group. In Figure 4.4 an example can be seen that was completed by C7 in the comparative group. In both examples the participants were able to complete the “Ten Block” challenge while following the listed constraints.

Figure 4.3

Ten Block Challenge Experimental Group



[redacted]

[redacted]

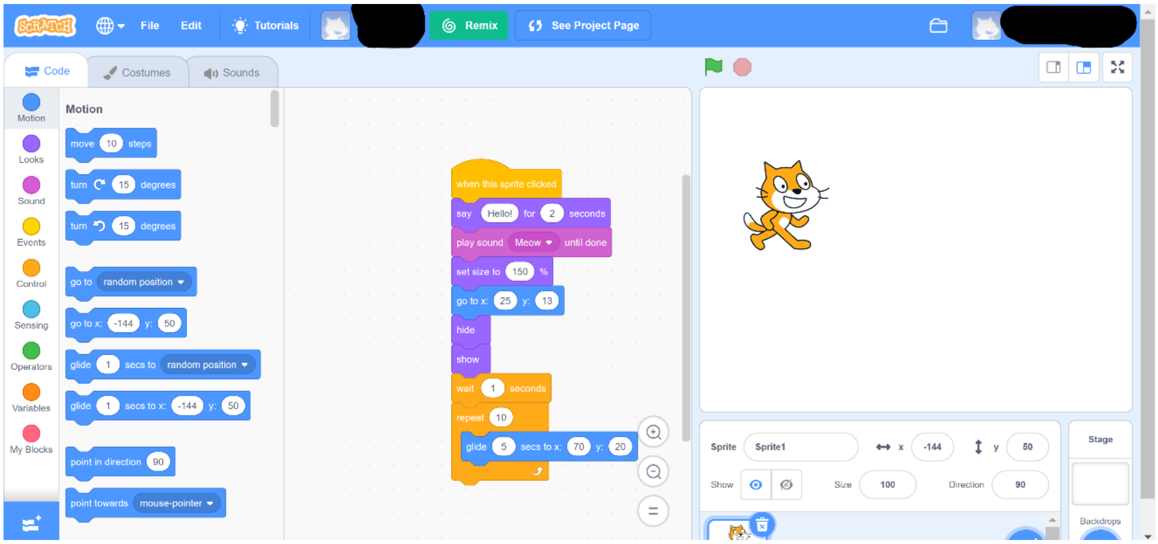
```
when this sprite clicked
repeat 10
    set size to 100 %
    play sound Meow ▾ until done
    glide 5 secs to x: 15 y: -10
    go to x: 10 y: -6
    say Hello! for 2 seconds
    hide
    wait 1 seconds
    show
```

Sprite Sprite1 x 12 y -8

Show Size 100 Direction 90

Figure 4.4

Ten Block Challenge Comparative Group



[redacted]

[redacted]

```
when this sprite clicked
say Hello! for 2 seconds
play sound Meow ▾ until done
set size to 150 %
go to x: 25 y: 13
hide
show
wait 1 seconds
repeat 10
    glide 5 secs to x: 70 y: 20
```

Sprite Sprite1 x -144 y 50

Show Size 100 Direction 90

The last components that made up the Algorithmic Thinking category was debugging. Debugging means to find and fix mistakes. Debugging was a daily occurrence because students were learning something new and were consistently fixing mistakes or altering designs.

In the Experimental group debugging also occurred when learning to cast-on. Since most of the students in the experimental group had never been exposed to knitting, they struggled to learn to get the yarn on the needle in a process called the cast-on. This process took double the time the teacher had allotted due to students learning to debug what they were doing incorrectly. In Figure 4.5 and Figure 4.6 E1's progress in learning to cast-on can be seen. In Figure 4.5 E1 is learning to cast-on and in Figure 4.6 E1 has debugged what they were doing wrong (pulling the yarn over) and their stitches look more correct.

Figure 4.5

E1 Learning to Cast on

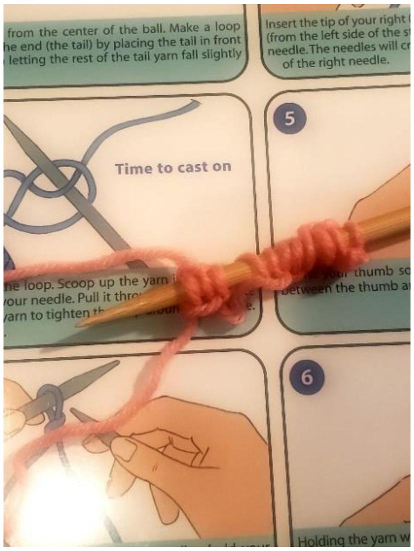
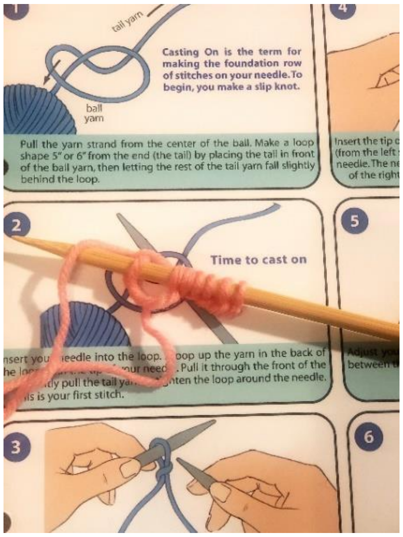

from the center of the ball. Make a loop
he end (the tail) by placing the tail in front
letting the rest of the tail yarn fall slightly
Insert the tip of your right
(from the left side of the s
needle. The needles will cr
of the right needle.
5
Time to cast on
he loop. Scoop up the yarn
your needle. Pull it thro
yarn to tighten
your thumb so
between the thumb a
6
Holding the yarn w

Figure 4.6

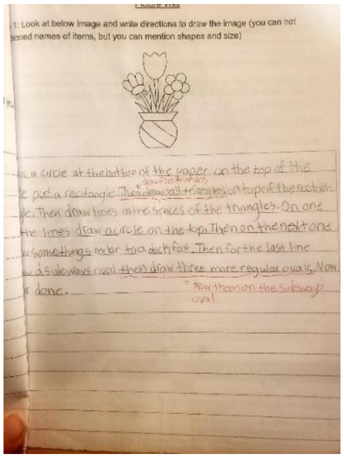
E1 Debugged Cast on


1
tail yarn
Casting On is the term for making the foundation row of stitches on your needle. To begin, you make a slip knot.
ball yarn
Pull the yarn strand from the center of the ball. Make a loop shape 5" or 6" from the end (the tail) by placing the tail in front of the ball yarn, then letting the rest of the tail yarn fall slightly behind the loop.
4
Insert the tip
(from the left
needle. The ne
of the right
2
Time to cast on
nsert you eedle into the loop. oop up the yarn in the back of
he loo our need . Pull it through the front of the
tly pull the tail ya nten the loop around the needle.
is is your first stitch.
5
Adjust you
between
3
6

Debugging can also be seen in the comparative group. In the “Picture This” activity (Figure 4.7), a coding “unplugged” activity, students were given a picture and they had to write down instruction on how to draw the picture. After they wrote their directions, they read the directions to a partner who drew what they heard. After looking at the end drawing, the student went back and debugged mistakes to make their instructions better. In the below image an example of this can be seen.

Figure 4.7

Picture This Activity



Picture This

1: Look at below image and write directions to draw the image (you can not ioned names of items, but you can mention shapes and size)

w a circle at the bottom of the paper on the top of the
le put a rectangle. Then draw tall triangles on top of the rectan-
gle. Then draw lines in the spaces of the triangles. On one
the lines draw a circle on the top. Then on the next one
w somethings similar to a duck foot. Then for the last line
d sideways oval then draw three more regular ovals. Now
r done.

draw 3 tall triangles

* draw them on the sideways oval

Overall, the Algorithmic Thinking category contained Algorithmic Thinking, abstraction, problem solving, decomposition, constraints, and debug. Examples of these processes can be observed in the research. Both the experimental and comparative group were able to demonstrate the different components, but neither group appeared to perform at a higher level based on this data.

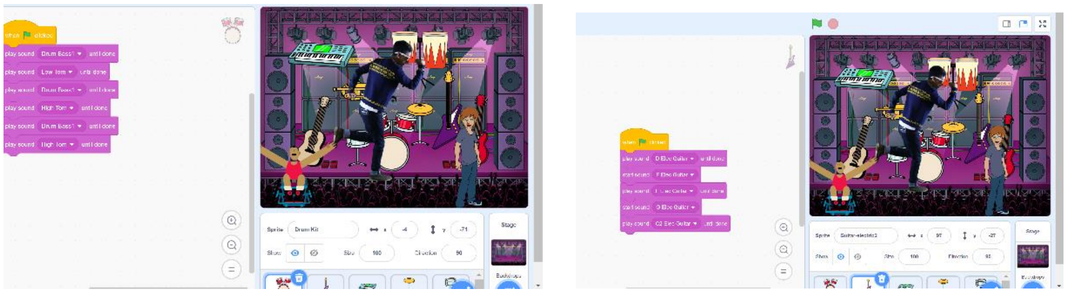
4.1.3 Student Examples of Parallelization Category

The second category that made up the Use of Computational Thinking Skills Theme was parallelization. Parallelization was developed from the parallelization, design, and loop codes. The parallelization category includes codes that enable the computer code to occur in working order with another computer code so multiple events can occur at the same time. To understand how the experimental and comparative group developed parallelization a closer examination of examples is necessary.

A simplified explanation is when multiple codes occur at the same time when signaled. This can most easily be seen in the “Build-a-Band” challenge on Scratch. Students were instructed to make a band and assign block code to each instrument. Some students made elaborate bands with multiple instruments and composed their own songs, while other students made a simpler band with fewer instruments and used pre-made sounds. In both cases there were examples of parallelization. In Figure 4.8 an example of parallelization in the “Build-a-Band” can be seen. This example was from the experimental group and shows how two instruments were performing music as the same time.

Figure 4.8

“Build-a-Band” Activity from Experimental Group



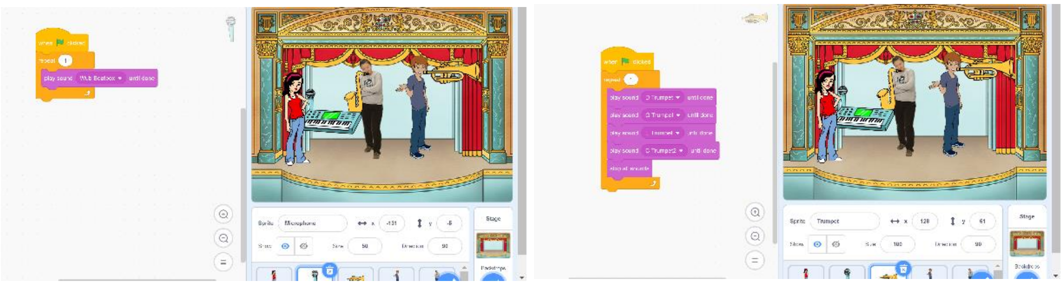
```
when ⚑ clicked
play sound Drum Bass1 until done
play sound Low Tom until done
play sound Drum Bass1 until done
play sound High Tom until done
play sound Drum Bass1 until done
play sound High Tom until done
```

```
when ⚑ clicked
play sound D Elec Guitar until done
start sound F Elec Guitar
play sound E Elec Guitar until done
start sound G Elec Guitar
play sound C2 Elec Guitar until done
```

Similarly, an example can also be seen from the comparative group in Figure 4.9. When looking at both examples, both groups have shown that they can compose different segments of code that performs at the same time when signaled.

Figure 4.9

"Build-a-Band" Activity from Comparative Group



```
when 🏴 clicked
repeat (1)
    play sound (Wub Beatbox ▾) until done
```

```
when 🏴 clicked
repeat ( )
    play sound (C Trumpet ▾) until done
    play sound (C Trumpet ▾) until done
    play sound (Trumpet ▾) until done
    play sound (C Trumpet2 ▾) until done
    stop all sounds
```

The second code that made up the parallelization category was design. Design, like the problem-solving category is very broad and as a result can be applied to many designs. Design was used daily for both groups when they were creating design in Scratch. In none of the challenges were students given step-by-step directions, so they had to design their project completely. Additionally, as seen in the examples of the "Build-a-Band" challenge, student also had to design multiple components to produce a product.

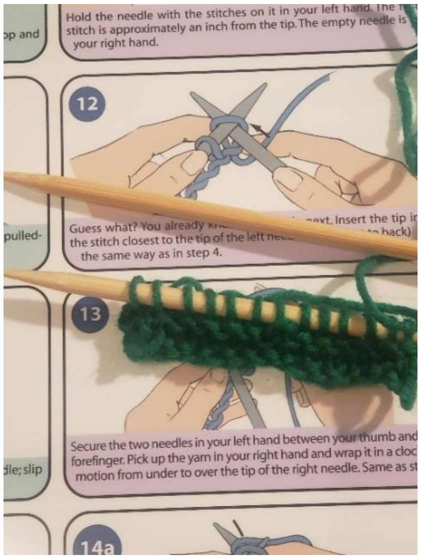
Loops are the final component that make up the Parallelization category. Loops can be seen in lesson five. Students were asked to define some examples of loops in their daily lives in their student journals. Case C3, in the comparative group, stated that walking was an example of a loop and case E2, in the experimental group, stated washing your hands was an example of a loop. In both cases students were able to determine that a loop is something that is repeated, but directions are not written over and over. Instead, the instructions are stated once and then you are told how many times to repeat.

In the experimental groups students learned this concept through knitting. After students learned the basic knit, they were introduced to repeats in patterns. Often in knitting, a set of steps is listed and then the instructions will say to repeat those steps a set number of times. This simplifies the instructions. During lesson five the researcher focused on teaching loops.

For the experimental group the researcher wrote "CO11, *turn work and K12* repeat * to * 4 times." This means that the students needed to cast-on eleven times and then turn their work and knit twelve times. The instructions went further and said to repeat the section between the asterisks, the turn work and knit twelve times, four times. An example of a student's work can be seen below in Figure 4.10.

Figure 4.10

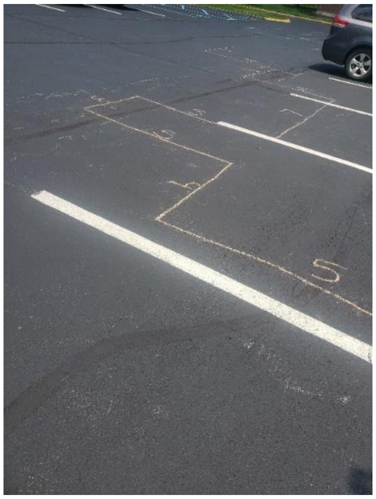
Example of "Loop" in Knitting


Hold the needle with the stitches on it in your left hand. The
stitch is approximately an inch from the tip. The empty needle is
your right hand.
op and
12
Guess what? You already
next. Insert the tip in
the stitch closest to the tip of the left
back)
the same way as in step 4.
pulled-
13
Secure the two needles in your left hand between your thumb and
forefinger. Pick up the yarn in your right hand and wrap it in a cloc
motion from under to over the tip of the right needle. Same as st
dle; slip
14a

Since the comparative group did not learn to knit, they had a different coding "unplugged" lesson. Students had to create a maze and write out instructions using loops such as describing how to take a step and then writing to repeat those directions a set amount of time. This can be seen in Figure 4.11 where a picture of the maze was drawn in both the experimental and comparative group, even though they experienced different lessons, students demonstrated their understanding of loops.

Figure 4.11

Example of Maze Loop Activity



Overall, the parallelization category contained parallelization, design, and loops. Examples of these processes can be observed in the research. Both the experimental and comparative group were able to demonstrate the different components, but neither group appeared to perform at a higher level based on this data.

4.1.4 Use of Computational Thinking Skills

Since Algorithmic Thinking and parallelization are both Computational Thinking skills it can be said that students were using Computational Thinking skills in their daily lessons during their instruction. The combination of these two themes developed the Computational Thinking skills theme. Based on student examples it can be observed that students are demonstrating these skills and the students that received the supplemental knitting instruction are performing at a similar level as those that received coding "unplugged" lessons.

Additionally, while not originally planned, the Bebras data was analyzed to be used as supportive evidence to compare of performance between the two groups. The data can be seen in Table 4.3. Two trends can be observed. First, both groups performed best on the first two challenges and performed the poorest on the last two challenges. The last two challenges required more math skills that students may not have possessed. Second, both groups performed very closely in the total average of correct answers. There was only one percentage difference between them. The combination of these two observations further supports the idea that computational skills were taught and that both groups performed at a similar level.

Table 4.3

Bebras Challenge Percentage of Correct Answers

	Bebras 1	Bebras 2	Bebras 3	Bebras 4	Bebras 5	Total Average Correct
Experimental						
Correct/Total	10/12	9/11	9/12	4/12	5/12	
Average	83%	82%	75%	33%	42%	63%
Comparative						
Correct/Total	10/11	8/11	6/9	3/8	6/11	
Average	91%	73%	67%	34%	55%	64%

4.1.5 Concluding Remarks Regarding Research Question One

In conclusion, to answer the research question, students that received knitting instruction developed or improved their computational skills. This can be seen in the student examples. Additionally, when the experimental group was compared against the comparative group, the experimental group performed at a similar level. This showed that using knitting to supplement instruction was comparable to the coding "unplugged" instruction. As a result, knitting can be used in instruction and can be expected to produce similar results as traditional coding "unplugged" instruction.

A counter argument could be made that the knitting had nothing to do with the computational skills and it was all due to the computer coding instruction. Additionally, students may have already possessed the Computational Thinking skills and the research did not know since there was no pre-assessment. If this study was repeated these would be areas to address. Additionally, in future studies, to gain a better understanding on the relationships between knitting and computation thinking, students could just be taught knitting and the impact on Computational Thinking skills could be observed.

4.2 Research Question Two: Computer Science Identity

During the literature review a defining element of the effectiveness of coding was an improvement in attitude towards coding and computer science (Bell & Vahrenhold, 2018). Taub et al. (2012) defined the attitudes of coding as "representing evaluations towards an "attitude object" in dimensions such as good/bad, harmful/beneficial, pleasant/unpleasant and likable/unlikable; for example, evaluating computer science as boring or tedious" (p. 8:5). Additionally, Taub et al. (2012) defined attitude to "include the motivational factors that influence a behavior" (p. 8:5). This includes the motivation to pursue a study of computer science.

4.2.1 Data Analysis

The theoretical framework introduced the idea of combining different topics, such as knitting and coding. The combination of these differing topics was referred to as the nexus of practice (Scollon, 2001). The nexus of practice, or more specifically, the nexus of STEAM, was the guiding Theoretical Framework of the Knitting Code program. The nexus of STEAM believes that the use of arts in STEM can promote a different type of learning that is more meaningful and inclusive (Peppler & Wohlwend, 2018). By knitting and coding the identities of both clash and produce a new identity that may be more inclusive and provide different access points for those that struggle with traditional coding instruction (Peppler & Wohlwend, 2018). This concept led to the second research question which is…

How does combining knitting and computer coding impact the identity of who studies computer science in adolescent students?

The data suggest that both groups received similar frequencies of use of identity as a code during first-cycle coding. This implies that the experimental group developed a similar identity as the comparative group as someone that could study computer science. Besides the identity code, additional coded data revealed the use of engaging, applicable, creativity, and communication. These codes were determined through the articles read during the literature review and development of the theoretical framework. Once specific point of interest was that the experimental group found the lessons more engaging compared to the comparative group. While this was not a specific research question, it was a finding that emerged from data analysis.

Once the first cycle coding process started these data codes were categorized into two main categories named Engagement and Identity. Engagement and Identity were chosen as the main category names over the other codes due to their frequency as codes in the data. The remaining codes that were related to identity development were then examined and placed into one of the two categories based on their purpose when coded in the data. This can be seen in Table 4.4.

Table 4.4

Codes to Categories

Codes	Categories	Definition/Explanation
Engaging, Applicable, Creativity	Engagement	Category includes components that keep students invested in what they are doing
Identity, Stereotypes, Communication	Identity	Category includes codes that shows student taking ownership for their projects.

After developing these two categories they were combined into one theme called the "I can be a computer scientist". The emergence of this theme shows that students were demonstration an identity towards computer science. This can be seen in Table 4.5.

Table 4.5

Categories to Themes

Categories	Themes	Explanation
Engagement Identity	"I can be a computer scientist"	Categories dealt with how the participants viewed themselves as able and willing to computer code and work with computers.

When comparing experimental group results against the comparative group, the experimental group received similar frequencies of coded data. This explains why different categories and themes were not developed for each group. This also suggests that the experimental group developed an identity similar to the comparative group. The exception is that the experimental group appeared to find the knitting instruction more engaging, but less applicable to their life. This may be due to their limited exposure to knitting. The frequencies of use of coded data can be seen in Figure 4.12 for the experimental group and Figure 4.13 for the comparative group.

Figure 4.12

Frequency of Codes for Experimental Group

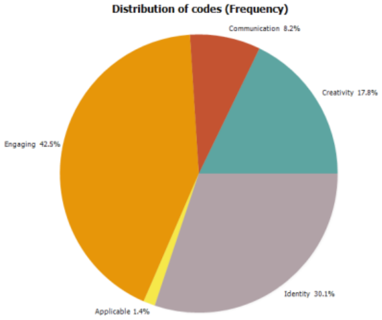
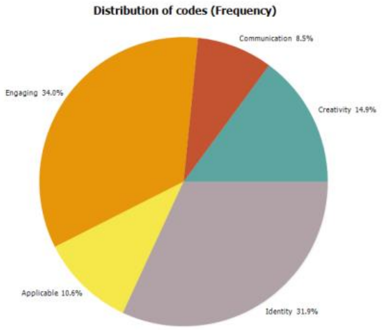


Figure 4.13

Frequency of Codes for Comparative Group



To get a more in-depth understanding of computer science identity development an examination of student work is needed.

4.2.2 Student Examples of Engagement Category

The engagement category was composed of the engaging, applicable, and creativity codes. This category includes components that kept students invested in what they were doing. What is unique about this category is that this category contains the biggest difference between the experimental and comparative group. In the experimental group there was more frequency of engagement codes and in the comparative group there was more frequency of applicable codes. It is debatable if this means one group developed more engagement overall since both of those codes fall under the engagement category. To gain a better understanding on how students developed engagement a closer examination of the encompassing codes needs to occur.

The engaging codes was used when students were enjoying and giving effort in what they were working on. This was most often noted in the daily classroom observations. The teacher would note amount of participation and quotes. Consistently the experimental groups had a higher level of participation ranging from 90%-100% while the experimental groups usually ranged from 70%-80%. Participation was determined by counting the students that were actively working on their project and not staring into space for over five minutes, asking to leave the room and being gone for over five minutes, or doing another activity for over five minutes. Additionally, for both groups, when participation was the lowest was usually during the knitting or “unplugged” instruction. Based on this, the computer coding instruction held about the same level of engagement, but the supplemental instruction of either knitting or “unplugged” coding discouraged some students.

Since there were more students less engaged in the comparative group it could be hypothesized that the “unplugged” instruction was not as engaging as the knitting instruction. To further support this claim the researcher recorded quotes in the daily observations. For the experimental group there was often complaining when knitting was not used such as E7 that repeatedly would ask every day when not knitting, “Why can’t we knit today?” For the comparative group there was more complaining when computers were not used. C10 asked, “Why are there no computers out today? Does this mean we have to write?” Taub et al. (2012) also experienced similar results with diminishing engagement with “unplugged” instruction. More research may be necessary to understand how “unplugged” instruction in any format impacts engagement in computer coding.

Besides the daily participation student engagement could be observed when it was time to end class and pack up. When the researcher would ask students to start getting ready to leave the students in the experimental group showed more hesitancy in switching classes. These conversations/observations were recorded in the daily notes. One conversation occurred after asking the experimental group to pack up at the conclusion of the “Build a Band” lesson.

Researcher: Please log off your Chromebook and get ready to leave

Case E3: Can I stay after class?

Researcher: Why?

Case E3: Because I want to keep working.

Researcher: I can’t keep you in my class and cause you to miss another class.

Case E3: Well, can I come to your room and work on it during lunch?

Researcher: No, I will be in the cafeteria monitoring, and no one will be in my classroom. You will have time tomorrow to work, what do you still need to do?

Case E3: My project is done, but I want to improve it more and try some things. Can I do this at home?

Researcher: Of course. You have your login information and can use a personal computer to access your account.

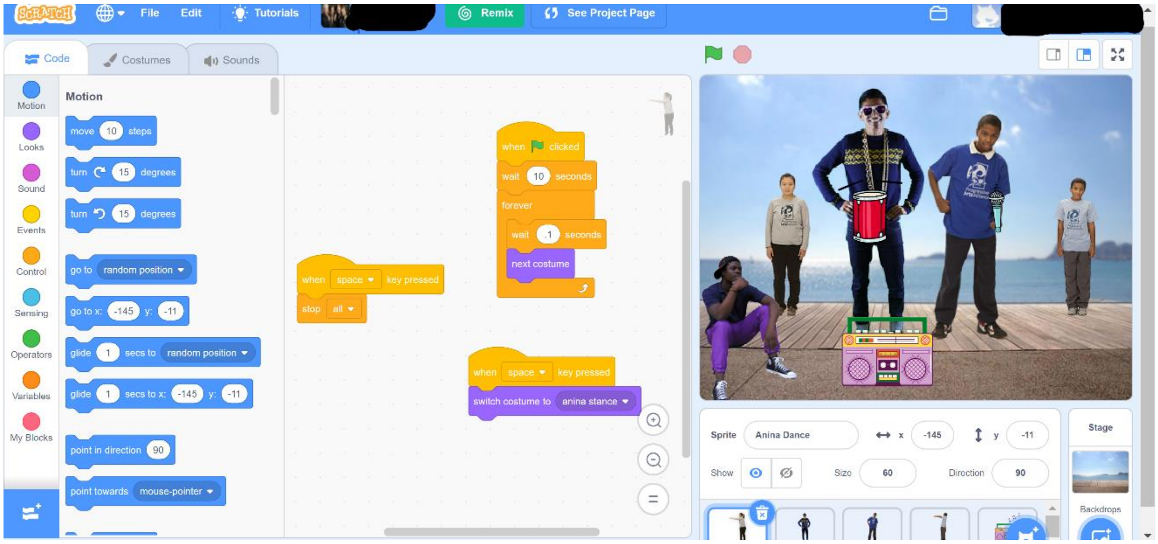
Additionally, besides using Scratch at home, some students in the experimental group wanted to take home their knitting needles to practice. The teacher did not allow this until the conclusion of the program because she was worried the materials would not return. While only a couple of students asked to take home their knitting needles or complete Scratch projects at home, there were no student in the comparative group that asked. This shows that students were more engaged and wanted to continue their instruction on their own time in the experimental group.

Applicable is the second code that developed the Engaging category. Applicable is defined as relating to student's lives. While it may appear that the knitting supplemental instruction was more engaging it did not appear to be as applicable to students. Evidence of this can also be seen in the daily observations. For the experimental groups the researcher heard statements such as, "How does this help with computer code?" and "People still do this?" In comparison, the comparative group used Tik Tok dances, that they had watched the night before, for their Lesson 1B dance activity.

The final code that completes the Engaging category is creativity. One observation that was recorded frequently in the daily observations were students' unique ways of completing the projects. Case E9 even stated, in an audiovisual interview, that one of their favorite parts of the class was the openness of the class to complete a project in a way they wanted. This can be seen in the cumulative "Music Video" challenge on Scratch. Students were instructed to use the skills they and learned through the course to create a music video with dancing and music. There were a variety of projects, and all were unique. This can be seen in Figure 4.14 (experimental group example) and Figure 4.15 (comparative group example).

Figure 4.14

Music Video Scratch Project for Experimental Group



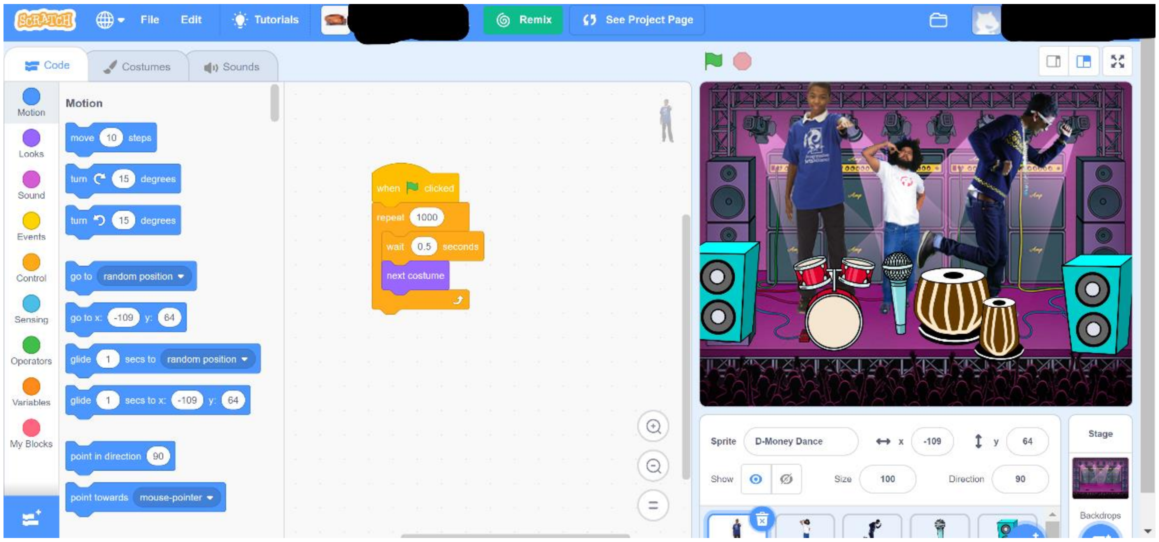
[redacted] [redacted]

```
when space key pressed
stop all

when flag clicked
wait 10 seconds
forever
    wait .1 seconds
    next costume

when space key pressed
switch costume to anina stance
```

Sprite: Anina Dance x: -145 y: -11
Show Size: 60 Direction: 90

Figure 4.15

Music Video Scratch Project for Comparative Group



[redacted] [redacted]

```
when flag clicked
repeat 1000
    wait 0.5 seconds
    next costume
```

Sprite: D-Money Dance x: -109 y: 64
Show Size: 100 Direction: 90

While not obvious in these pictures, both projects included unique combinations of characters, movement, and music. In the experimental group and the comparative group music videos students used different costumes within the video. This was not designated as something that was necessary, but students wanted their sprites to change with he music and fit the style.

The Engaging category was a combination of the engaging, applicable, and creativity codes. Overall, the experimental group showed more engagement while the comparative group showed more application. Both groups showed similar creativity with the experimental group being slightly higher. Some questions that arose was if coding “unplugged” activities were as engaging as computer coding and if engagement outweighs lack of application to a students’ life.

4.2.3 Student Examples of Identity Category

The Identity category was composed of the identity, stereotypes, and communication data codes. Overall, the identity category includes codes that shows students taking ownership for their projects and seeing themselves as a computer scientist. For this study, since identity and stereotypes were so closely related the researcher tended to code the data as identity and as a result not much data was coded as stereotypes. As a result, final data was combined. Additionally, both the identity and communication frequency of codes was similar for both groups.

To get a better understanding of the identity codes the CSAIS survey was used. The survey was a tool developed to evaluate students’ identity towards computer science. All students, in both groups, were given this survey at the beginning and end of the three-week sessions. The data was analyzed using the Wilcoxon Signed Rank Test due to the matched samples from the pre- and post- survey. The results revealed any changes in identity towards computer science from the beginning of the session to the end. The null hypothesis was there was no change in attitude from the pre- to post- survey and the alternative hypothesis was there would be a change in identity from the pre- to post-survey. The test was conducted at a five percent significance and in both cases the test statistic of the experimental (128) and comparative (117) group was less than the critical value of 137 which means the null hypothesis was rejected. This means that there is sufficient evidence to suggest that for both the experimental and comparative groups there was a significant positive change in identity towards computer science from the beginning to the end of the three-week session.

One point of interest in the CSAIS survey was the fourteenth question which asked, “The challenge of solving problems using computer science appeals to me”. This question showed the greatest improvement in the experimental group while in the comparative group the data did not change. While the overall results showed similar growth, question fourteen stuck out as an outlier and shows that students in the experimental group found using computer science to solve problem more appealing from the start of the session to the end while there was no growth in the comparative group.

Additionally, while the survey showed overall growth in identity in both groups there were incidents with individuals where the identity for certain questions decreased. These decreases were small and not enough to impact the overall results, but it should be noted that not every student showed growth in identity on every question.

Another piece of data that supports students developing more if an identity towards computer science can be seen in the researchers' daily notes. At the beginning of the study the researcher wrote for both groups that students "were not excited to be at summer school" or "student laid their head down on the desk". By the end of the session the researcher's results had greatly improved. Notes the researcher wrote included, "a parent contacted me to tell me how much their child is enjoying this class", "teachers are reporting that students are taking about my class", "students are asking to stay in my class and not go to the next", and "students are asking if I will not delete the Scratch class so they can get Scratch after the session". These comments show a progression from students not wanting to be in the class to enjoying what they are doing and talking about the course outside of class. This was present in both groups.

The CSAIS survey was also used to determine students' ideas about stereotypes. Section three of the survey had to do with gender constructs and section four dealt with professional constructs. In both groups there was minimal change in the gender constructs from the pre- to post- survey, but the pre- survey showed positive stereotypes that women and men can be computer scientist. Therefore, since the results were already positive and there was minimal change, it can be determined that students believe that men and women can be equally successful in computer science. In comparison the professional construct section did show positive growth for both groups of students. This means that students' perceptions as who studies computer science and if it is something they could do improved from the beginning of the course to the end of the course of both groups. This included viewing a computer scientist as someone that had additional interests and friends compared to the beginning when students thought computer scientist were anti-social and only worked with computers.

Finally, communication was present in both groups. Students were eager to share their designs with the teacher and their classmates. Additionally, at the end of the three-week session, the researcher had students complete a gallery walk where students observed each other's projects and provided feedback. This can be seen in Figure 4.16 and Figure 4.17. For both groups communication was consistent, and the researcher did not observe or document any differences between the two groups.

Figure 4.16

Gallery Walk of Music Video Scratch Projects

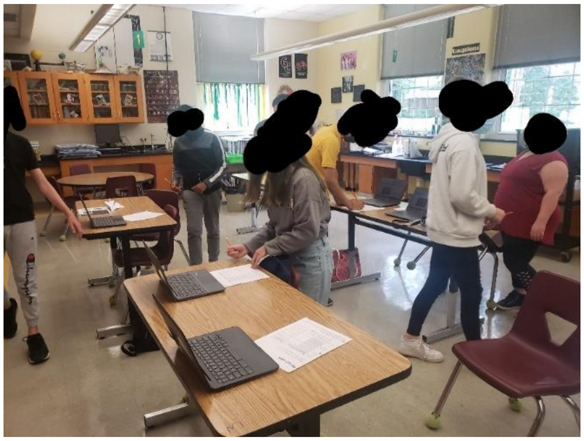
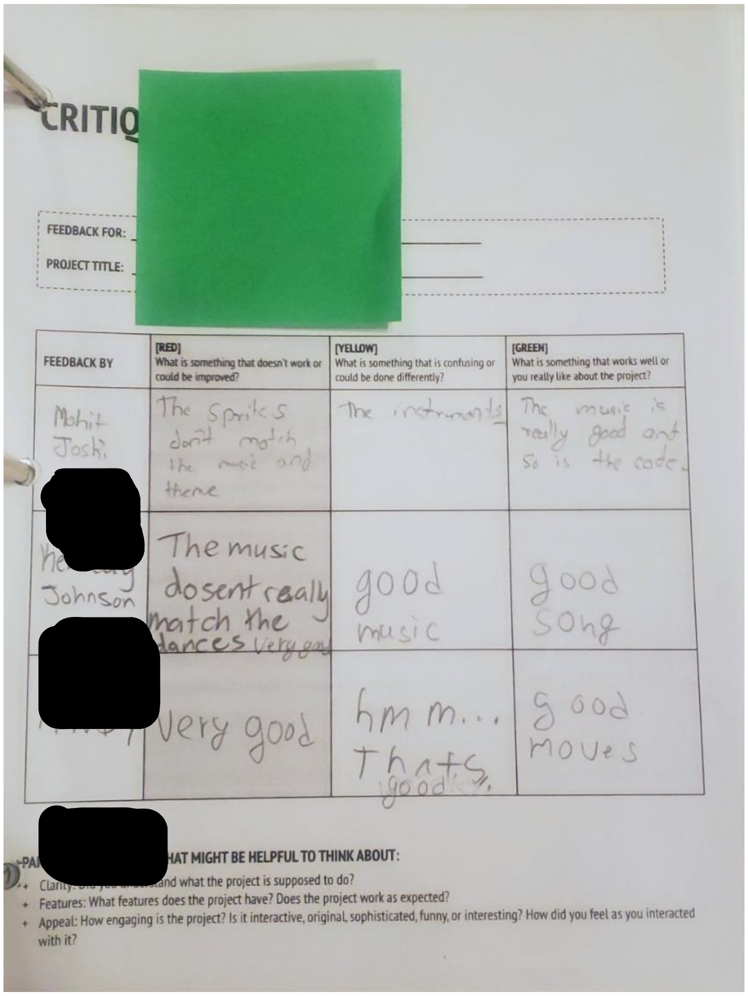


Figure 4.17

Feedback from Students on Music Video Scratch Project



CRITIQ [redacted]

FEEDBACK FOR: [redacted]

PROJECT TITLE:

FEEDBACK BY	[RED] What is something that doesn't work or could be improved?	[YELLOW] What is something that is confusing or could be done differently?	[GREEN] What is something that works well or you really like about the project?
Mohit Joshi	The Sprites don't match the music and theme	The instruments	The music is really good and so is the code.
he [redacted] Johnson	The music dosent really match the dances Very good	good music	good song
[redacted]	Very good	hmm... Thats, good	good moves

PA [redacted] HAT MIGHT BE HELPFUL TO THINK ABOUT:

- Clarity. and what the project is supposed to do?
- Features: What features does the project have? Does the project work as expected?
- Appeal: How engaging is the project? Is it interactive, original, sophisticated, funny, or interesting? How did you feel as you interacted with it?

4.2.4 "I Can be a Computer Scientist" Theme

The overall theme that emerged when combining the Engagement and Identity categories was the "I can be a Computer Scientist" theme. This theme combined categories dealing with how the participants viewed themselves as able and willing to computer code and work with computers. The data, while differing between groups" from the Engagement category showed that students were engaged with the computer coding but questioned if the "unplugged" coding component was actually valuable and if application outweighed engagement. These questions could not be answered based on this study alone. In the Identity category, the data showed that there was a change in identity towards computer science based on the CSAIS Survey data. The results provided evidence that students were able to view themselves as someone that might study computer science more at the conclusion of the session than at the beginning. This was consistent for both the experimental group and comparative group.

4.2.5 Connection to Theoretical Framework

Going back to the Theoretical Framework and the Nexus of Practice some statements can be made. First, as a reminder, the Nexus of Practice is the combination of different practices that form a nexus through links and relationships (Hui et al., 2017). As people develop practices, they develop certain abilities that signify membership in a group (Hui et al., 2017). Practices can become organized into a nexus (Hui et al., 2017). Learning, both passive and active, is necessary to form practices, and as a result signals membership to a group (Hui et al., 2017). Additionally, learning occurs when different nexus intersects, because knowledge is shared through the interaction (Hui et al., 2017). Conflicts can also occur when nexus converge because they may not share practices, and as a result, old practices can develop into new practices (Wohlwend, 2014). The new practices are not what form the change, instead how the practices interconnect and display a new organization of practices is what causes a change (Hui et al., 2017). This is referred to as a disruption of practice (Scollon, 2001). Key practices from the different nexus may form a new nexus and provide an invitation for new membership of that nexus that combines key practices and values from the combined nexus (Medina & Wohlwend, 2014). When nexus combine, the new nexus can change practices and identities that would be slow to change (Wohlwend, 2014). “When new practices emerge in nexus, the results can be transformative, allowing greater access and broader participation” (Peppler & Wohlwend, 2018, p. 91).

When applying the Nexus of Practice to this study the practices can be defined as the different components of computer science and computer coding. The combination of these practices determines the Nexus of Computer Science. If a student struggles to develop these practices, then they will struggle to feel like they belong to the Nexus of Computer Science. Additionally, if a student struggled with the idea that they could be a computer scientist they could be introduced to computer science through a different Nexus. In this case the other Nexus was knitting. When the Nexus converge, such as the Knitting Code” program a new way of feeling like a member of the Nexus emerges. In other words, if a person does not feel like they could be a computer scientist, but they feel like they could knit when the two meet that feeling of belonging could transfer.

Based on the results of this study, students did develop more of an identity as a computer scientist and someone that could computer code. Thus, students felt more of a membership to the Nexus of computer science. Additionally, the knitting allowed students to approach the Nexus of computer science using an alternative path. Data did not suggest that the experimental group developed more of an identity than the experimental, but the research question focused on if the Knitting Code program was comparable to more traditional computer coding instruction. You could argue that since both groups developed an identity towards computer science the knitting was not necessary. While knitting itself may not be necessary, the research shows that it is possible to use alternative Nexus to teach computer science in hopes that an individual who does not feel like a member of computer science may have an alternative path to reach this identity.

4.2.6 Concluding Remarks for Research Question Two

In conclusion, to answer the research question, students that received knitting instruction did show growth in identity towards someone who could be a computer scientist. This can be seen in the student examples and survey results. Additionally, when the experimental group was compared against the comparative group, the experimental group performed at a similar level. This showed that using knitting to supplement instruction produced similar computer science identities and shows that knitting instruction was comparable to the coding "unplugged" instruction.

A counter argument could be that using Scratch helped develop identity, not the knitting "unplugged" coding. If this study was repeated this could be an area to address. A study could be conducted using a traditional coding language and Scratch. The change in identity between these two groups could be investigated to determine how Scratch impacts identity.

4.3 Summary

In conclusion from the data four categories were developed which were then developed into two themes. These two themes, Use of Computational Thinking Skills and "I Can be a Computer Scientist", helped answer the two research questions. Since this is a case study the how and why should be answered for each question.

Research question one questioned how learning to knit while learning to code facilitated the understanding of Computational Thinking skills. To answer the how, learning to knit while learning to code helped facilitate the learning of Computational Thinking skills such as abstraction, Algorithmic Thinking, and parallelization. This can be seen through the student examples above. To answer the why, while students completed the Knitting Code program they had to use the Computational Thinking skills to answer the Bebras questions, develop Scratch projects, and to learn to knit/read knitting patterns.

Research questions two questioned how combining knitting and coding impacted identity and stereotypes of who studies computer science. In terms of how, the Knitting Code program improved students' identity towards seeing themselves as someone that could study computer science. In term of why, it is unclear if it is the combination of different Nexus of the use of Scratch, but the use of these programs allowed students to see themselves as someone who could study computer science. This can be seen in student examples of survey results.

Overall, the study answered the research questions, but new questions arose from the research. Additionally, the purpose of a case study is to suggest implications about a larger phenomenon (Yin, 2003, p 144). The purpose of this case study was to explore the idea if knitting could be helpful when teaching coding. In conclusion, based on data and the research questions, it is possibly to imply that learning computer coding through knitting is comparable to learning computer coding using coding “unplugged” instruction.

Chapter 5 IMPLICATIONS AND SIGNIFICANCE

5.1 Conclusion

Computer coding is, “the process of identifying and labeling each step to complete a task” (Lee & Junoh, 2019, p. 712). There is an increase in demand for those that known how to computer code and as a result colleges need to produce a diverse group of leaners to fill this demand (Ehrlinger et al., 2018; Varma, 2006). Computer “coding skills should be considered among basic skills, and they are of equivalent importance as reading” (Tonbuloglu & Tonbuloglu, 2019, p. 404).

To develop an interest in computer science and computer coding for children a variety on tools have been developed. One tool is called computer coding “unplugged” and teaches the skills of computer coding without using technology (Lee & Junoh, 2019). Another tool are drag-and-drop programs, such as Scratch, which have been developed to make coding more scaffolded and engaging (Resnick et al., 2009).

Another growing idea is that knitting can be used as a tool to teach computer coding. Knitting and coding have similarities, and as a result there is a growing idea that students that learn to knit will be better computer coders (Roberts, 2019). There have been numerous claims that knitting can teach the concepts of computer coding, but these claims have little scientific backing (Buckner, 2015; Roberts, 2019). This is where the idea of the Knitting Code program emerged.

The Knitting Code program combined a variety of tools including STEAM, block-based computer coding, and “unplugged” coding. STEAM is the combination of art and STEM. This is an example of the nexus theory because there is a combination of contradictory fields. By combining both, the learner was challenged to use creative design elements to solve problems. Additionally, the “Knitting Code” program used drag-and-drop programs so that the material was accessible to adolescent students. The goal of drag-and-drop was to introduce students to coding in a friendly manner. Drag-and-drop will combine the aspects of art and computer coding.

“Unplugged” coding will also be taught through knitting. This “unplugged” component will allow students to learn to knit and then knit their code. Since “unplugged” coding was never intended to be taught independently, the lessons will be combined with a drag-and-drop coding-based curriculum. The nexus theory is guiding this research due to the combination of art and coding. The different components found through the literature review have shaped the goals of the “Knitting Code” program.

This case study was guided by two research questions based off the emerging idea that knitting and coding are similar. Due to this similarity, statements have been made that knitting can promote computer coding skills (Buckner, 2015; Roberts, 2019). This study was set up as an exploratory case study design due to its focus on answering “how” and “why” questions in a hard to control scenario using a modern phenomenon (Yin, 2003, p.1). Overall, this case study wanted to increase understanding between the relationship between knitting and computational skills (Yin, 2003). Due to this focus a multi-case design was needed that also included a comparative group used for theoretical replication. Additionally, while the total number of participants was smaller than other forms of experimental study design, the data is still valid because case studies are focused more on how individual cases are impacted over time and less concerned with frequencies or incidents (Yin, 2003).

The case study occurred during two, three-week, sessions where each case received a total off fourteen, forty-eight-minute lessons. Additionally, the study occurred in a public middle school during a summer enrichment program. The researcher taught four classes each session. Two of the classes received computer coding instruction and knitting instruction. This was the experimental group. The other two classes received computer coding instruction and coding "unplugged" instruction. This was the comparative group. The teacher-researcher designed the curriculum based on the "Creative Computing" curriculum by the Harvard Graduate School of Education (2019), her own knowledge of knitting, and with the help of a content expert in computer science.

The data for this case study came from a variety of sources and was analyzed using qualitative data coding. Data included student journals, class observations, identity surveys, audiovisual interviews based on Bebras Challenges, and student artifacts. Multiple sources of data were used to build validity in the case study design and to discourage bias (Yin, 2003). Before the study started precoding was conducted using deductive methods to develop a list of potential code words. During first cycle coding the data was analyzed using descriptive and In Vivo coding (Saldana, 2021). From that data four categories emerged that were named (1) Algorithmic Thinking (2) Engagement (3) Parallelization (4) Identity. During second cycle coding pattern coding was used. As a result of pattern coding two themes emerged that were called (1) Use of Computational Thinking Skills (2) "I can be a computer scientist". The end goal of a case study is to generalize what was discovered, not particularize (Yin, 2003, p11). As a result, some trends were discovered regarding the original research questions.

The first research question asked how learning to knit while learning to code facilitated the understanding of Computational Thinking skills including abstraction, Algorithmic Thinking, and understanding of parallelization in adolescent students. Data suggested that students that received knitting instruction developed or improved their computational skills. Additionally, when the experimental group was compared against the comparative group, the experimental group performed at a similar level. This showed that using knitting to supplement instruction was comparable to the coding “unplugged” instruction.

The second research question asked how combining knitting and coding impacted identity and stereotypes of who studies computer science. To answer the second research question, students that received knitting instruction did show growth in identity towards someone who could be a computer scientist. This can be seen in the student examples and survey results. Additionally, when the experimental group was compared against the comparative group, the experimental group performed at a similar level. This showed that using knitting to supplement instruction contributed to similar identity development and shows that knitting instruction was comparable to the coding “unplugged” instruction.

5.2 Implications

The purpose of case studies is to suggest implications for a larger phenomenon (Yin, 2003). The purpose of this study was to determine if knitting could supplement computer coding instruction. Based on this case study, the data suggests that knitting could be an alternative way to teach coding so that the disruption of Nexus could occur and allow more students to enter the field of computer science. What this implies is that knitting could be a potential teaching method for computer coding. This case study proved that the Knitting Code curriculum had the same success as more traditional "unplugged" methods. On a larger scale, this means that alternative methods to teaching computer science, and other disciplines, should be considered as potential pedagogical approaches. By studying the two cases the researcher was able to support the phenomenon that by combining seemingly unrelated Nexus students can be successful and be provided with a different pathway to access the content.

Due to constraints such as time and state teaching mandates using knitting may not be possible to assist with teaching computer coding. What can be taken away from this study is when developing instruction for teaching computer coding alternative methods should be considered. These alternative methods could be beneficial in terms of students learning Computational Thinking skills and developing an identity. Knitting may not be the best option since it requires materials, times, and expertise. Additionally, knitting showed to not be applicable to students' lives.

Potential alternative methods of instruction for all content should take into account the Nexus of Practice. Instructions should consider trying to disrupt the Nexus of their content by combining seemingly unrelated fields. This disruption may help students understand concepts in a different way and this disruption may allow other students to enter the Nexus of their content. Instructors should take away from this study that there is not one correct way to teach or learn content. If that instructor is struggling to teach, engage, or bring in new students they should consider what are some students' interests.

By learning more about students' interests the instructor could become more familiar with the Nexus of that topic. The instructor could then create instruction combining the Nexus of their content with the Nexus of the students' interests. By doing this the instructor may create a pathway for students to access the content in a way that is more interesting and familiar to themselves.

In conclusion, the Knitting Code curriculum was successful. The study was trying to determine if teaching computer coding through knitting would be as successful as more traditional computer coding "unplugged" instruction. Success was measured based on understanding of computation thinking skills and identity development. While data analysis showed that both cases had similar success, students in the experimental group that received knitting instruction were more engaged. This is an important piece of data because, while the overall study results did not change, students in the experimental group were enjoying learning about the content more. This could be important when considering future studies and use in the classroom. Overall, the disruption of Nexus appeared to benefit students because both groups were equally successful and the group that received knitting instruction found the content more engaging. Therefore, the students in the experimental group may have a greater chance to study computer science in college and fill the demand in computer science careers.

5.3 Significance/Future Research

Based on results from this case study several future studies could be developed. First, since knitting was not applicable to students' lives, research could be conducted on how other art/crafts, such as origami, could be used to teach computer coding. Second, to gain a better understanding on the relationships between knitting and computation thinking, students could just be taught knitting and the impact on Computational Thinking skills could be observed. Third, a study could be conducted using a traditional coding language versus Scratch. The change in identity between these two groups could be investigated to determine how Scratch impacts identity and development of Computational Thinking skills. Forth, interviews could be conducted with STEM professionals and see if they do a craft/art for free time and compare against another professions. Fifth, future studies could focus on minorities and see how using knitting impacts minorities' identities. Finally, this same study could be repeated, but instead of using coding and knitting instruction, just knitting instruction could be used for the experimental group.

Overall, this case study examined how learning to knit while learning to computer code impacted learning of Computational Thinking skills and computer science identity. Results showed, that while this method is not better than computer coding instruction with computer "unplugged" instruction, it is comparable. This is significant, because these results answer the question, addressed in chapter one, that knitting can teach computer coding. While there is no evidence to suggest using knitting is more beneficial, the fact it can be used provides options for students that may not typically study computer science. Additionally, this option may allow teachers to provide alternative teaching methods, which as a result may bring in more students to fill the need for more computer scientists.

APPENDICES

APPENDIX A

Curriculum Breakdown

Title Of Lesson	**Preparation (Lesson 0)**
Length of Lesson	One Class Periods
Standard(s) Taught	N/A
Overall Theme	Identity
Problem Based Learning Level	Introduction of Problem
Purpose of Lesson	This will be the first day of class and the students and researcher/teacher will meet each other for the first time, go over the consent/assent, receive Chromebooks, set up their Scratch account, and complete an identity survey
Overview of Lesson	1. The teacher will assign seat and take attendance. 2. The teacher will explain the purpose of the class, the research, and go over the consent/assent forms. 3. To reduce bias and limit any power play from the teacher the students may feel the teacher/researcher will leave the room as students sign their form. Another teacher will collect these forms and keep them safe until the conclusion of the experiment. 4. Afterwards, the teacher will give students the identity questionnaire. 5. Once students have completed the questionnaire, they will be shown how to log into their Scratch classroom account. 6. Students will be given time to log into their account and explore Scratch 7. At the conclusion of class students will be introduced to the problem: Students have been told they need to make a music video for their favorite band, but due to

	Covid no one can meet in person, so instead they need to create a music video using code/Scratch.
End Goal	At the end of the lesson students will have filled out their assent, taken their identity survey, logged into Scratch, and been introduced to the guiding problem.
Materials Needed	Computer, Scratch Class Account, Parent Consent Forms, Student Assent Forms, CSAIS Questionnaire
Teacher Questions	- What are some of your initial ideas about how to create a music video? - Has anyone ever used Scratch? What was your experience? - What do you think of when you heard computer science or computer coding? - Who do you envision works with computers for their job?
Potential Student Questions	- Why do I need to learn this? - Do I have to assent? - Will I not have to do the work if I don't assent? - How do I log into Scratch? - What does _________ question mean?
Differentiation	Read to the student questions on the survey and what is written on the assent form
Assessment (journal)	N/A
Assessment (video)	N/A
Title Of Lesson	**Introduction to Scratch (Lesson 1)**
Length of Lesson	- Two Class Periods
Standard(s) Taught	KY CS Standards: M-AP-02, M-AP-04 CSTA Standards: 1A-AP-08, 1B-AP-08 K12CS Practice 5: Creating Computational Artifacts K12CS Concepts: Computing Systems, Algorithms and Computer coding
Overall Theme	Sequence/Algorithm and Constraints

Problem Based Learning Level	Inquiry
Purpose of Lesson	In the previous lesson students were presented with the problem of needing to create a music video using Scratch. During this lesson students will be introduced to Scratch by completing a tutorial and creating their first project.
Overview of Lesson (plugged)	1. Parent consent form will be collected by a different teacher and kept safe. 2. Students will receive their folder and place their daily journal sheets in the folder. 3. Students sign into their classroom Scratch accounts. 4. Students will complete the assigned "Getting Started with Scratch" tutorial to learn how to use the Scratch interface. 5. As students finish students will be allowed to experiment with motion, sprites, looks, costumes, sounds, and backdrops. 6. Students will be given a challenge to create a project of their choosing using 10 designated blocks (go to, glide, say, show, hide, set size to, play sound until done, when this sprite clicked, wait, and repeat). 7. Give students time to share their projects with their peers and a class discussion will occur using the "potential teacher questions" as a guide. 8. As students are working, they will fill in their project journal
End Goal	At the end of the lesson students should be able make a cat sprite dance by completing the tutorial and create a project by only being allowed to use 10 designated blocks.
Materials Needed	Computer, Scratch Class Account, Step by Step Handout, 10 Blocks Handout, Folders, Journal Sheets
Potential Teacher Questions	- What was surprising about this activity? - How will this lesson help you create your end music video?

	- Why is sequence important when specifying instructions and provide an example? - How did having constraints make you think of things differently?
Potential Student Questions	- How do I _____________? - What does ________ block do? - Do I have to use all 10 blocks? - Can I use the same block multiple times? - Why can I only use those 10 blocks?
Supplemental Lesson (“unplugged”)	Control Group 1. Students will be taught what sequence and an algorithm are. 2. Students will complete the Bebras Challenge 3. Students will watch a compilation of popular Tik Tok dances. 4. Each student will privately choose one of the dances shown. 5. The students will attempt to write detailed instruction on how to complete their chosen dance. 6. After, the teacher will ask for volunteers and one student will read their instructions while another student follows the directions. 7. The class will try to guess which dance the student wrote the directions for. 8. There will be a class discussion using the “potential teacher questions” as a guide. 9. As students are working, they will be pulled into the hall to complete their audiovisual interview. Experimental 1. Students will be taught what sequence and an algorithm are. 2. Students will solve the Bebras Challenge 3. Students will be given knitting materials and explained their purpose. 4. Students will be taught to cast on and make the basic knit. 5. Students will be given time to practice

	6. While students are practicing the teacher will lead a discussion about why steps are important when learning these stitches and use the “potential teacher questions” as a guide 7. As students are working, they will be pulled into the hall to complete their audiovisual interview
End Goal	At the end of the lesson students should learn the importance of detailed steps, why the order or steps matter, and what an algorithm is.
Materials Needed	Computer, Projector, Notebook Paper, YouTube, Knitting Bags with Prepared Materials
Teacher Questions	- What was frustrating about this activity? - Why are detailed steps important? - When creating steps, why is the order or sequence important? - How is an algorithm connected to sequence and steps? - Why is a math problem called an algorithm?
Potential Student Questions	- How many steps are needed? - Why won’t _______ move like I want? - Why is this step confusing?
Interventions	- Have student complete the tutorial again and them observe the sample projects before creating their own. Also, watch the Scratch “how-to” video on YouTube
Differentiation	- Work with a partner, teach left-handed knitting, modeling
Enrichment	Control – Show the “Making a PB&J sandwich video. Experimental – Show other crafts that require similar steps/sequence as knitting
Assessment (journal)	- How did it feel to be led step-by-step through the activity? - When do you feel most creative? - What was hard/easy about only be able to use 10 blocks?
Assessment (video)	- Bebras Challenge: Animation - Explain/Show what was learned from “unplugged” lesson and how it relates to sequence and algorithms

Title Of Lesson	**Debugging Code (Lesson 2)**
Length of Lesson	- Two Class Periods
Standard(s) Taught	KY CS Standards: M-AP-02, M-AP-04, M-AP-10 CSTA Standards: 1A-AP-08, 1B-AP-08 K12CS Practice 5: Creating Computational Artifacts K12CS Concepts: Computing Systems, Algorithms and Computer coding
Overall Theme	Sequence/Algorithm and Debugging
Problem Based Learning Level	Reflect
Purpose of Lesson	Now that students are becoming familiar with scratch it is important for them to be able to reflect on a project and determine if there are mistakes. Today's focus will be on identifying and correcting mistakes.
Overview of Lesson (plugged)	1. Parent consent form will be collected by a different teacher and kept safe. 2. Folders will be passed out and journal sheets will be placed in the folder. 3. Complete any discussions/correct misconceptions from previous lesson. 4. Have students get on their class Scratch account and complete "Debug" 1.1-1.5 programs (as time permits) 5. Afterwards there will be a class discussion using the "potential teacher questions." 6. As students complete their assignments, they will fill in their project journal.
End Goal	At the end of the lesson students should have investigated some problems and found a solution (debug) and had the idea of sequence reenforced
Materials Needed	Computer, Scratch Class Account, "Debug It" Handout, Folders, Journal Sheets
Teacher Questions	- What was one of the problems you debugged and how did you fix the problem?

	- Did other have alternative approaches to fixing the problem?
Potential Student Questions	- What is the problem? - How do I solve the problem? - Why does debugging matter?
Supplemental Lesson (“unplugged”)	Control Group 1. Teacher will review what sequence and an algorithm is. 2. Students will complete Bebras Challenge 3. Students will be given different pictures 4. Students will list a steps to reproduce the image 5. Students will then read their directions to their partner and have them draw the image. 6. Once the picture is drawn the original image and the drawn image will be compared. 7. Students will identify mistakes in the images and “Debug” their directions 8. There will be a class discussion using the “Potential Teacher Questions” 9. While students are working, they will be pulled into the hall to complete their audiovisual interview. Experimental 1. Teacher will review what sequence and an algorithm is. 2. Students will complete Bebras Challenge 3. Students will be given knitting materials 4. Students will review casting on and knitting 5. Students will be taught the purl stitch 6. A discussion will be held talking about “debugging” knitting mistakes and using the “Potential Teacher Questions.” 10. While students are working, they will be pulled into the hall to complete their audiovisual interview.
End Goal	At the end of the lesson students should learn the importance of detailed steps and why debugging

	is needed to correct mistakes and improve upon design.
Materials Needed	Computer, Projector, Paper, Pictures, Knitting Bags with Prepared Materials, Folders
Teacher Questions	- What was frustrating about this activity? - Why are detailed steps important? - Is debugging only for correcting or can it also improve a design? - What is an example of debugging that occurred?
Potential Student Questions	- Why won't ________ draw the image like I said? - How is knitting like sequence? - Why does debugging matter?
Interventions	- List the mistakes before trying to solve
Differentiation	- Work in groups of 2-4 people to debug
Enrichment	Control – provide a more detailed image to create directions for. Experimental – Show knitting patterns and discuss the "code" of the pattern
Assessment (journal)	- What was the problem? - How did you identify the problem? - How did you fix the problem?
Assessment (video)	- Bebras Challenge: Animation - Explain/Show what was learned from "unplugged" lesson and how it relates to sequence and debugging
Title Of Lesson	**About Me Collage (Lesson 3)**
Length of Lesson	- Two Class Periods
Standard(s) Taught	KY CS Standards: M-AP-02, M-AP-04 CSTA Standards: 1A-AP-08, 1B-AP-08 K12CS Practice 5: Creating Computational Artifacts K12CS Concepts: Computing Systems, Algorithms and Computer coding
Overall Theme	Sequence/Algorithm

Problem Based Learning Level	Inquiry and Communicate
Purpose of Lesson	Today students will create a collage showing information about the band they will be creating a music video for. You need to communicate why they think this band is the best.
Overview of Lesson (plugged)	1. Parent consent form will be collected by a different teacher and kept safe. 2. Folders will be passed out and journal sheets will be placed in the folder. 3. Complete any discussions/correct misconceptions from previous lesson. 4. Have students get on their class Scratch account. 5. The teacher will show examples of interactive collages 6. Have students create an "about the band" interactive Scratch collage 7. Have a gallery walk and have students fill out the feedback form (if this is not completed during day 1, this will be done during the beginning of day 2). 8. Afterwards there will be a class discussion using the "potential teacher questions." 9. As students complete their assignments, they will fill in their project journal.
End Goal	At the end of the lesson students will become more familiar with a variety of Scratch blocks and create an open-ended Scratch project
Materials Needed	Computer, Projector, Scratch Class Account, "About Me" Handout, Critique Sheet, Folders, Journal Sheets
Teacher Questions	- What are you most proud of, why? - What inspired you? - What would you want to do next?
Potential Student Questions	- How do I do _______? - How do I solve _____ problem? - I want my sprite to do _______ what code do I need to use?
Supplemental Lesson ("unplugged")	Control Group 1. Teacher will review what sequence and an algorithm is.

	2. Students will complete Bebras Challenge 3. Complete gallery walk/discussion from yesterday 4. Have students go to “Explore” and search for different types of projects. 5. Share a neat project with a neighbor. 6. There will be a class discussion using the “Potential Teacher Questions.” 7. While students are working, they will be pulled into the hall to complete their audiovisual interview. Experimental 1. Teacher will review what sequence and an algorithm is. 2. Students will complete Bebras Challenge 3. Complete gallery walk/discussion from yesterday 4. Students will be shown a knitting pattern 5. Afterwards students will be introduced to the website “Ravelry” and be shown different knitting patterns. 6. A discussion will be held talking about “debugging” knitting mistakes and using the “Potential Teacher Questions.” 7. While students are working, they will be pulled into the hall to complete their audiovisual interview.
End Goal	At the end of the lesson students should see the variety of options of projects that can be made using Scratch or using knitting
Materials Needed	Computer, Projector, Paper, Pictures, Knitting Bags with Prepared Materials, Folders
Teacher Questions	- What strategies did you use to find interesting projects? - How might each example project help with future work?
Potential Student Questions	- Is there a project about ____? - How did the person come up with ____ idea?
Interventions	- Provide a list of blocks to use
Differentiation	- Partner-pair, modeling

Enrichment	Control – provide a more detailed image to create directions for. Experimental – Show knitting patterns and discuss the "code" of the pattern
Assessment (journal)	- What did you get stuck on? How did you get unstuck? - What did you discover from looking at others' About My Band projects?
Assessment (video)	- Bebras Challenge - Explain/Show what was learned from "unplugged" lesson and how it relates to sequence
Title Of Lesson	**Build-A-Band (Lesson 4)**
Length of Lesson	- Two Class Periods
Standard(s) Taught	KY CS Standards: M-AP-02, M-AP-04 CSTA Standards: 1A-AP-08, 1B-AP-08 K12CS Practice 5: Creating Computational Artifacts K12CS Concepts: Computing Systems, Algorithms and Computer coding
Overall Theme	Events/Parallelism
Problem Based Learning Level	Inquiry and Communication
Purpose of Lesson	Today students will start working on creating their music video by creating their band while incorporating the sounds and instruments the band uses.
Overview of Lesson (plugged)	1. Parent consent form will be collected by a different teacher and kept safe. 2. Folders will be passed out and journal sheets will be placed in the folder. 3. Complete any discussions/correct misconceptions from previous lesson. 4. Have students get on their class Scratch account. 5. The teacher will introduce events and loops and show examples of them in example Build-A-Bands

	6. Have students build a band by creating a sprite, incorporating music, and making music instruments interactive. 7. (If time) Have a gallery walk and have students fill out the feedback form (if this is not completed during day 1, this will be done during the beginning of day 2). 8. Afterwards there will be a class discussion using the “potential teacher questions.” 9. As students complete their assignments, they will fill in their project journal.
End Goal	At the end of the lesson students will have created a program with interactives sprites and different sounds
Materials Needed	Computer, Projector, Scratch Class Account, “Build-A-Band” Handout, Critique Sheet, Folders, Journal Sheets
Teacher Questions	- What was challenging? - Did you make what you envisioned? - Was there something you could not figure out? Could someone else figure this out?
Potential Student Questions	- How do I do _______? - How do I solve _____ problem? - I want my sprite to do _______ what code do I need to use? - How do I add sound? - Can I upload my own music?
Supplemental Lesson (“unplugged”)	Control Group 1. Teacher will review what events and parallelism are. 2. Students will complete Bebras Challenge 3. Complete gallery walk/discussion from yesterday 4. Students will work in pairs to program their partner to complete a simple task (walk across the room). 5. They will then “reset” their partner and add a parallel task (talk while walking across the room). 6. Then two groups will work together to get their partners to interact during an event

	7. Groups will have a chance to share their work. 8. There will be a class discussion using the "Potential Teacher Questions." 9. While students are working, they will be pulled into the hall to complete their audiovisual interview. Experimental 1. Teacher will review what events and parallelism are. 2. Students will complete Bebras Challenge 3. Complete gallery walk/discussion from yesterday 4. Students will be shown a knitting pattern and be shown events and parallelism present in the pattern. 5. Afterwards students will be given time to start their project (scarf). 6. A discussion will be held using the "Potential Teacher Questions." 7. While students are working, they will be pulled into the hall to complete their audiovisual interview.
End Goal	At the end of the lesson students should start to understand events and parallelism and explain how they apply to coding.
Materials Needed	Computer, Projector, Paper, Pictures, Knitting Bags with Prepared Materials, Knitting Pattern, Folders
Teacher Questions	- What is an event? - How is parallelism? - What were different ways actions were triggered between two groups that caused them to interact?
Potential Student Questions	- Can I have my partner do ______? - What is an example of an event? - How do I make my partner stop/start one action while continuing another?
Interventions	- Provide a list of possible commands
Differentiation	- Partner-pair, modeling

Enrichment	Control – encourage groups to add three or more actions at one ad events to stop/start these actions Experimental – discuss examples of events and parallelism in different crafts
Assessment (journal)	- What did you do first? - What did you do next? - What did you do last?
Assessment (video)	- Bebras Challenge - Explain/Show what was learned from "unplugged" lesson and how it relates to events and parallelism
Title Of Lesson	**It's Alive (Lesson 5)**
Length of Lesson	- Two Class Periods
Standard(s) Taught	KY CS Standards: M-AP-02, M-AP-04, M-AP-07 CSTA Standards: 1A-AP-08, 1B-AP-08 K12CS Practice 5: Creating Computational Artifacts K12CS Concepts: Computing Systems, Algorithms and Computer coding
Overall Theme	Events/Parallelism, Loops
Problem Based Learning Level	Inquiry
Purpose of Lesson	Today students will be introduced to the idea of a loop to simplify codes and make their sprites appear to move.
Overview of Lesson (plugged)	1. Parent consent form will be collected by a different teacher and kept safe. 2. Folders will be passed out and journal sheets will be placed in the folder. 3. Complete any discussions/correct misconceptions from previous lesson. 4. Have students get on their class Scratch account. 5. The teacher will discuss loops and show examples of animation as looping 6. Have students create animation by choosing a sprite, adding a costume, and

	adding loops to the code to make the sprite “come alive” 7. Afterwards there will be a class discussion using the “potential teacher questions.” 8. As students complete their assignments, they will fill in their project journal.
End Goal	At the end of the lesson students will become familiar with sequence, loops, parallelism, and events.
Materials Needed	Computer, Projector, Scratch Class Account, “It’s Alive” Handout, Folders, Journal Sheets
Teacher Questions	- What did you make? - How did you use a loop to create animation? - How could loops simplify code?
Potential Student Questions	- How many times can I make something loop? - Why won’t by sprite stop/start the loop? - Can a loop be used for ______?
Supplemental Lesson (“unplugged”)	Control Group 1. Teacher will review loops and events/parallelism are. 2. Students will complete Bebras Challenge 3. Students will work with a partner to create a list of steps for their blindfolded partner to complete an obstacle course. 4. After they succeed, students will use loops to create the fewest number of steps possible. 5. If time remains groups will have a chance to race through the course using their code. 6. There will be a class discussion using the “Potential Teacher Questions.” 7. While students are working, they will be pulled into the hall to complete their audiovisual interview. Experimental 1. Teacher will review what loops and events/parallelism are.

	2. Students will complete a Bebras Challenge 3. Students will be shown a knitting pattern and have to identify examples of loops in the pattern. 4. Afterwards students will be given time to continue working on their project (scarf). 5. A discussion will be held using the "Potential Teacher Questions." 6. While students are working, they will be pulled into the hall to complete their audiovisual interview.
End Goal	At the end of the lesson students should understand loops simplify a pattern making it easier to write and to read
Materials Needed	Computer, Projector, Paper, Pictures, Knitting Bags with Prepared Materials, Knitting Pattern for Loops, Knitting Patterns for Project, Folders
Teacher Questions	- Are loops easy to use? - Why would someone want to write a loop instead of writing all the code? - Why would someone want to read a loop instead of reading all the code?
Potential Student Questions	- Can a loop be used for _________? - How many times can I use a loop? - Can a loop be put in a loop?
Interventions	- Have a student sketch their design on paper
Differentiation	- Partner-pair, modeling
Enrichment	Control – encourage students to make a loop within a loop Experimental – Show how loops are used a variety of crafts
Assessment (journal)	- What is animation? - List three ways you experience loops in real life (e.g., getting ready in the morning)
Assessment (video)	- Bebras Challenge - Explain/Show what was learned from "unplugged" lesson and how it relates to loops

Title Of Lesson	**Music Video (Lesson 6)**
Length of Lesson	- Two Class Periods
Standard(s) Taught	KY CS Standards: M-AP-02, M-AP-04, M-AP-07, M-AP-10 CSTA Standards: 1A-AP-08, 1B-AP-08 K12CS Practice 5: Creating Computational Artifacts K12CS Concepts: Computing Systems, Algorithms and Computer coding
Overall Theme	Events/Parallelism, Loops, Sequence/Algorithms, Debugging
Problem Based Learning Level	Inquiry
Purpose of Lesson	Today students will create their music video. Students’ creativity is the only limit. Students will use what they have learned to create a music video that they will be ready to share.
Overview of Lesson (plugged)	1. Parent consent form will be collected by a different teacher and kept safe. 2. Folders will be passed out and journal sheets will be placed in the folder. 3. Complete any discussions/correct misconceptions from previous lesson. 4. Have students get on their class Scratch account. 5. The teacher will introduce the idea of creating a music video in Scratch that uses concepts learned by providing examples 6. Students will then be given two days to create a music video that must include sound, an animated sprite, and interaction between the music and sprite 7. A brief discussion will be held about giving credit to the artist to prevent plagiarism 8. Afterwards there will be a class discussion using the “potential teacher questions.”

	9. As students complete their assignments, they will fill in their project journal.
End Goal	At the end of the lesson students will become familiar with sequence, loops, parallelism, events, algorithms, and debugging. Each student will have created a music video that they will be ready to share.
Materials Needed	Computer, Projector, Scratch Class Account, “Music Video” Handout, Folders, Journal Sheets
Teacher Questions	- What did you make? - What was easy? - What was hard? - What concepts learned did you use in your design?
Potential Student Questions	- Can I use ________? - How do I make ______ and _______interact? - Do I have to use all the concepts learned?
Interventions	- Provide a list of helpful code blocks
Differentiation	- Partner-pair, modeling
Enrichment	- Have students play computit.com and solve coding problems
Assessment (journal)	- What was a challenge you overcame? How did you overcome it? - What is something you still want to figure out?
Assessment (video)	- N/A
Title Of Lesson	**Conclusion**
Length of Lesson	One Class Periods
Standard(s) Taught	N/A
Overall Theme	Identity
Problem Based Learning Level	Communication
Purpose of Lesson	This will be the final day of class. Students will first get a chance to share their music video and then complete the CSAIS identify survey again.
Overview of Lesson	1. Students will be given time to share their music videos during a gallery walk

	2. Students will fill out a critique sheet over each other's projects. 3. Afterwards, the teacher will give students the identity questionnaire again. 4. Once students have completed the questionnaire, a final class discussion will be held using the "Potential Teacher Questions"
End Goal	At the end of the lesson students will have shared their music video and taken their final identity survey.
Materials Needed	Computer, Scratch Class Account, Critique Forms, CSAIS Questionnaire
Teacher Questions	- What were some projects you liked and why? - Was there something you didn't do that you wish you could have? - What was frustrating? - What did you enjoy? - How do you feel about coding? - Would you take another class about coding if offered?
Potential Student Questions	- How did _________ do _______? - What skills did _____ use?
Differentiation	Read to the student questions on the survey and what is written on the assent form
Assessment (journal)	N/A
Assessment (video)	N/A

APPENDIX B

SEMI-STRUCTURED OBSERVATION FORM

Lesson:

Date of Observation:

Time/Period:

Control or Experimental:

Description of Activity:

Computational Skills Taught:

General Observations:

Learner’s Response to Activity:

Strengths and Difficulties:

Students’ Comments/Quotes:
Students’ use of Computational Thinking Skills:
Identity Development or Alienation:
Preliminary Coding Themes:

APPENDIX C

CSAIS IDENTITY/ATTITUDE SURVEY

Part 1: Confidence Construct (taken from CSAIS Survey)

	Strongly Disagree	Disagree	Agree	Strongly Agree
I am comfortable with learning computing concepts.	1	2	3	4
I have little self-confidence when it comes to computing courses.	1	2	3	4
I do not think that I can learn to understand computing concepts.	1	2	3	4
I can learn to understand computing concepts.	1	2	3	4
I can achieve good grades (C or better) in computing courses.	1	2	3	4
I am confident that I can solve problems by using computer applications.	1	2	3	4
I doubt that I can solve problems by using computer applications	1	2	3	4

Part 2: Interest Construct (taken from CSAIS Survey)

	Strongly Disagree	Disagree	Agree	Strongly Agree
I would not take additional computer science courses if I were given the opportunity.	1	2	3	4
I think computer science is boring.	1	2	3	4
I hope that my future career will require the use of computer science concepts.	1	2	3	4
The challenge of solving problems using computer science does not appeal to me.	1	2	3	4

I like to use computer science to solve problems	1	2	3	4
I do not like using computer science to solve problems.	1	2	3	4
The challenge of solving problems using computer science appeals to me.	1	2	3	4
I hope that I can find a career that does not require the use of computer science concepts.	1	2	3	4
I think computer science is interesting.	1	2	3	4
I would voluntarily take additional computer science courses if I were given the opportunity.	1	2	3	4

Part 3: Gender Construct (taken from CSAIS Survey)

	Strongly Disagree	Disagree	Agree	Strongly Agree
I doubt that a woman could excel in computing courses.	1	2	3	4
Men are more capable that women at solving computing problems.	1	2	3	4
Computing is an appropriate subject for both men and women to study.	1	2	3	4
It is not appropriate for women to study computing.	1	2	3	4
Men produce higher quality work in computing that women.	1	2	3	4
Men are more likely to excel in careers that involve computing that women are.	1	2	3	4
Women produce the same quality work in computing as men.	1	2	3	4

Men and women are equally capable of solving computer problems.	1	2	3	4
Men and women can both excel in computing courses.	1	2	3	4

Part 4: Professional Construct (taken from CSAIS Survey)

	Strongly Disagree	Disagree	Agree	Strongly Agree
A student who performs well in computer science will probably not have a life outside of computers.	1	2	3	4
A student who performs well in computer science is likely to have a life outside of computers.	1	2	3	4
Students who are skilled at computer science are less popular than other students.	1	2	3	4
Students who are skilled at computer science are just as popular as other students.	1	2	3	4

Part 5: Demographics

Age: ______________

Sex: ______________

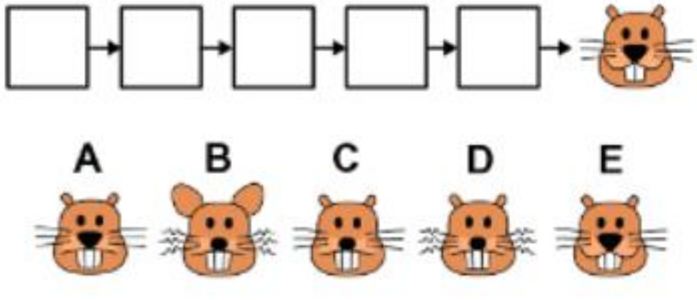
Race: _____________

APPENDIX D

BEBRAS COMPUTATIONAL THINKING CHALLENGES

Animation

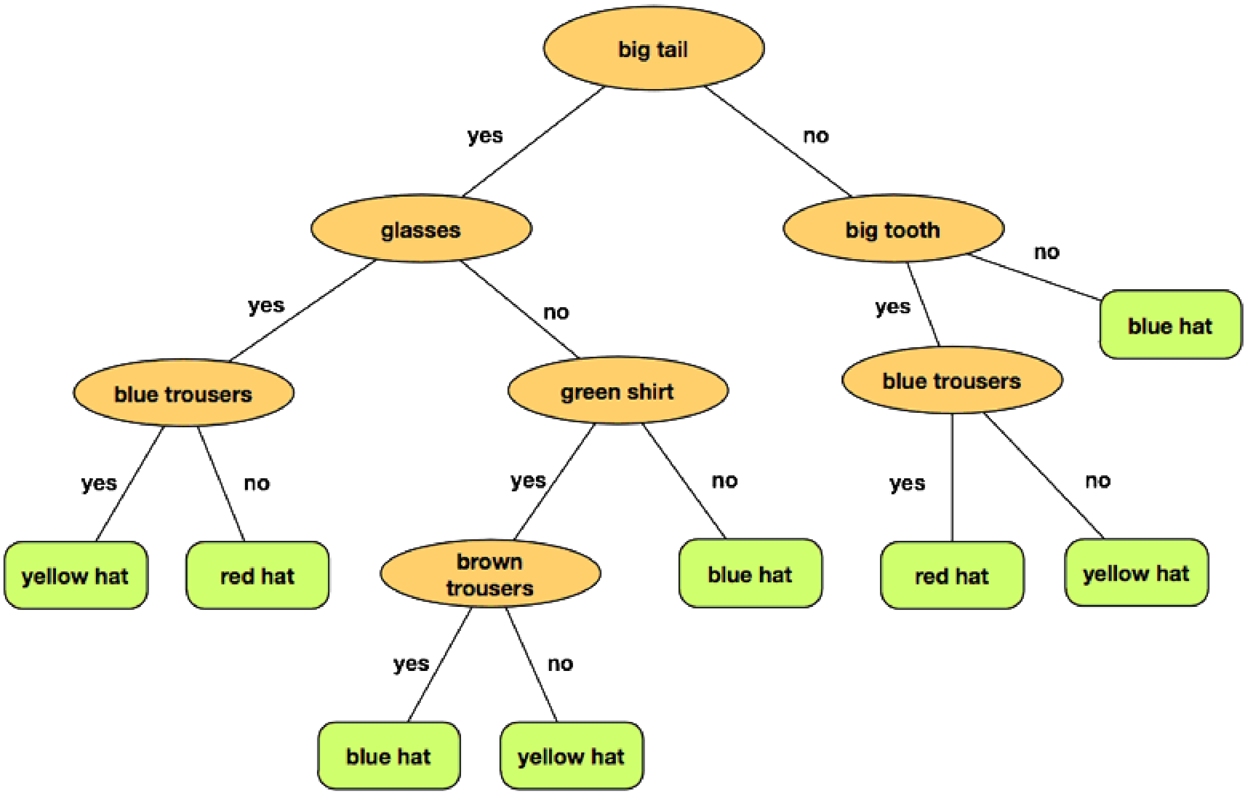
B-taro is planning an animation, which shows a sequence of pictures of a face. The animation should run smoothly. Therefore, the order of the pictures is correct, if only one attribute of the face changes from one picture to the next. Unfortunately, the pictures got mixed up. Now B-taro must find the correct order again. Luckily, he knows which picture is last. He labels the five other pictures with letters A to E.



What is the correct order of the five other pictures?
(1) D → B → E → C → A
(2) C → B → D → A → E
(3) D → B → C → E → A
(4) B → D → C → A → E

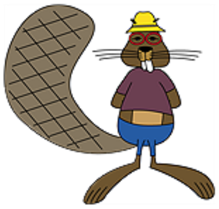
Dress Code for Beavers

Beavers like complex rule systems and have therefore established a new dress code. Some beavers don't use the correct combination of clothes. Use the graph to determine which beaver is dressed correctly. The graphic is called a tree because there is a single root node (the topmost) with branches connecting other nodes – like a real tree. At every node you have to decide which direction you want to go within the tree, you can’t go up again.
Which beaver is not dressed like the dress code?

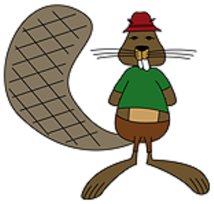


Answer:

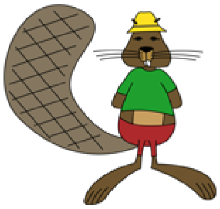
A



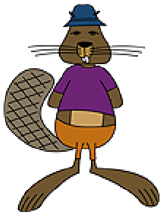
B



C

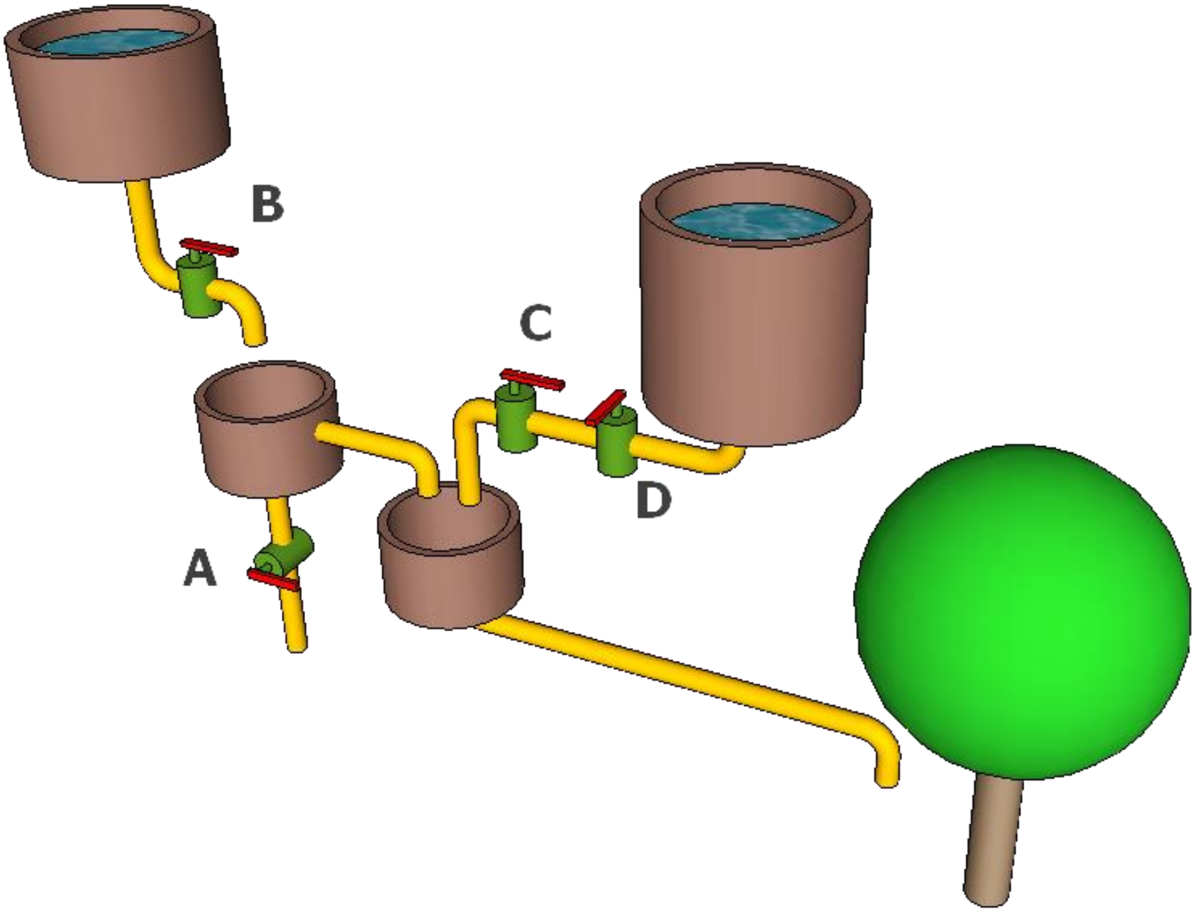


D



<u>Water Supply</u>

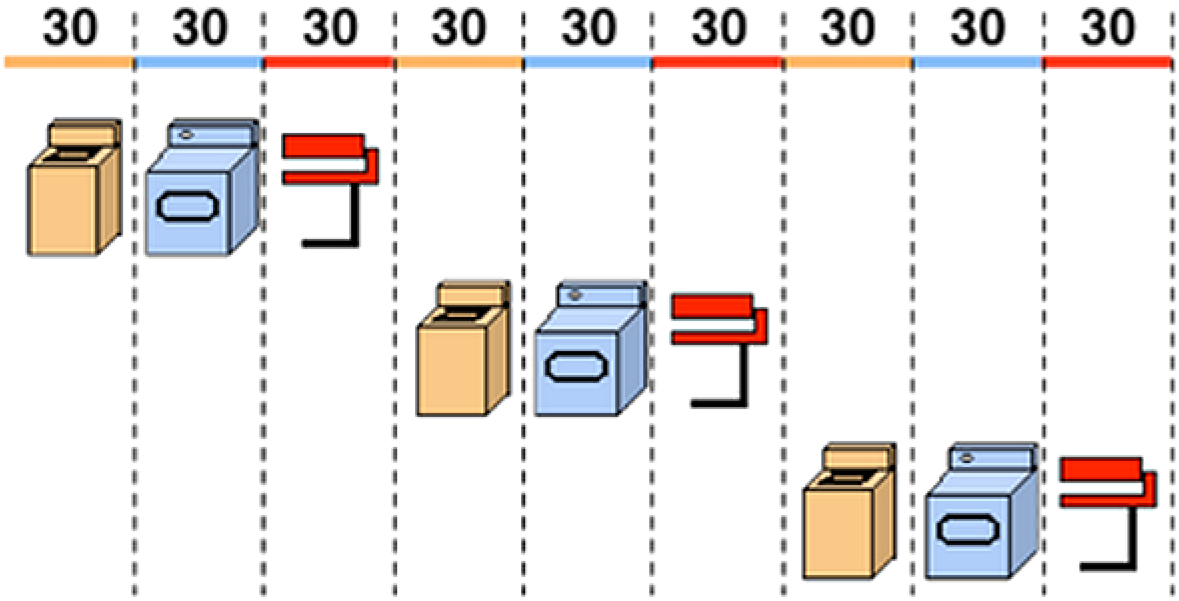
Beaver has constructed a pipeline system to water his apple tree.
The expressions contain variables A, B, C, D, which may be true or false. A variable has the value true, if the corresponding gate is open, and false, if it is closed.
In which case the apple tree gets water?



Answer:
A: A = false, B = true, C = false, D = false
B: A = true, B = true, C = false, D = false
C: A = true, B = false, C = false, D = true
D: A = false, B = false, C = false, D = true

Fast Laundry

Beaver Joe has started a new laundry business. He has got three machines: a washer, a dryer, and a pressing iron. Every machine is connected through its own timer which provides for half an hour of electricity.
So, when a client arrives, he needs 90 minutes for all of the three procedures. And three clients using the machinery consequently need 270 minutes.

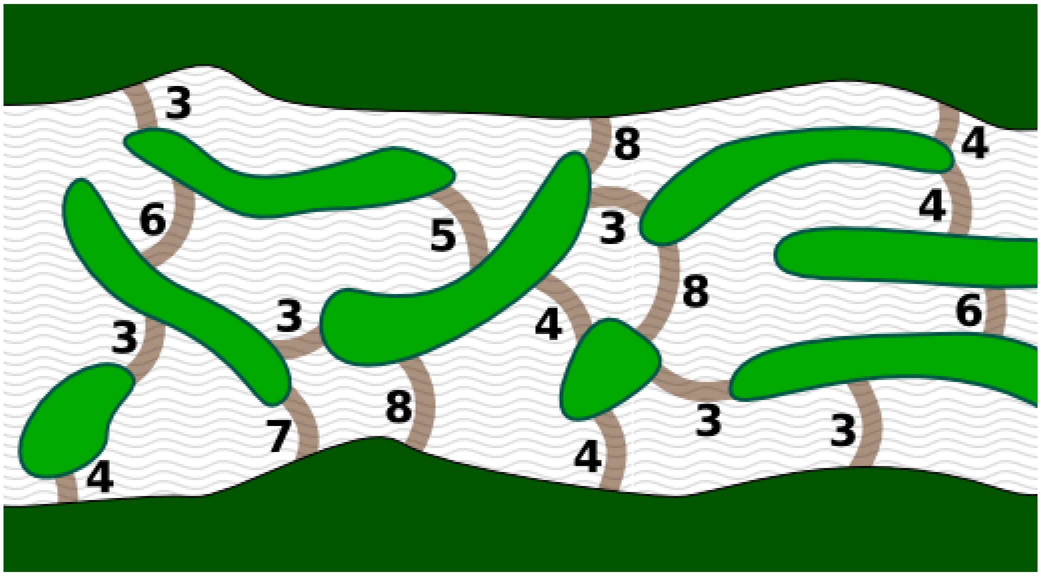


But now, there are three beavers arriving which are really busy. Each one them has enough clothes for a load of its own. But they agree that they want to finish as quickly as possible.
How many minutes does it take for all three of them to finish their laundry?

A) 90 minutes
B) 120 minutes
C) 150 minutes
D) 270 minutes

Beaver dam

The beaver community designs a new dam on the river. They want to use the least number of logs. They are smart, so they want to take advantage of the small islands in the river. The picture shows the river, the islands, and the number of logs needed to build each dam segment.



What is the least number of logs needed for the new dam?
1. 14 logs
2. 15 logs
3. 16 logs
4. 17 logs

BIBLIOGRAPHY

Adkins, S. J., Rock, R. K., Morris, J. J. (2017). Interdisciplinary STEM education reform: dishing out art in microbiology labratory. *FEMS Microbiology Letters*, 365(1), 1-8. https://doi- org.ezproxy.uky.edu/10.1093/femsle/fnx245

Adlington, L. (2015). Stitches in time: The story of the clothes we wear. *Cornerstone Digital*.

Atalay, B. (2011). Math and the mona lisa: The art and science of Leonardo da vinci. *Smithsonian Institute*, Washington, DC.

Bell, T. & Vahrenhold, J. (2018). CS unplugged: How is it used, and does it work? Adventures Between Lower Bounds and Higher Altitudes. *Lecture Notes in Computer Science*, (11011). https://doi.org/10.1007/978-3-319-98355-4_29

Bell, T. C., Witten, I. H., & Fellows, M. (1998). Computer Science Unplugged: Off-line activities and games for all ages. *Computer Science Unplugged*. http://csunplugged.org

Bozkurt, A., Ucar, H., Durak, G., & Iden, S. (2018). The current state of the art in STEM research: A systematic review study. *Cypriot Journal of Educational Sciences*, 14(3), 374-383. https://doi.org/10.18844/cjes.v14i3.3447

Brandreth, G. D. (1899). Writing secret codes and sending hidden messages. *Sterling Pub Co* Inc. https://www.amazon.com/Writing-Secret-Sending-Hidden-Messages/dp/0806963069

Buckner, K. (2015). How knitting is like coding. *Medium*.

Bush, S. B., & Cook, K. L. (2019). *Step into steam: Your standards-based action plan for deepening mathematics and science learning, grades K-5*. Corwin.

Bush, J. B., Gilmore, M. R., & Miller, S. B. (2020). Drag and drop programming experiences and equity: Analysis of a large-scale middle school s tudent motivation survey. *SIGSE* (20).

Butler, J. (1990). Gender trouble: Feminism and the subversion of identity. Routledge.

Campbell, C., & Walsh, C. (2019). Introducing the "new" digital literacy of coding in the early years. Practical Literacy: The Early and Primary Years, 22(3), 10-12. https://www.alea.edu.au/documents/item/1672

Chouliaraki, L., & Fairclough, N. (1999). Discourse in late modernity: Rethinking critical discourse analysis. Edinburgh University Press

Code.org (2020). CS fundamentals: Express. Retrieved from https://studio.code.org/s/express-2019

Code.org. (n.d.) Infographic Source Data. https://code.org/promote

Colucci-Gray, Laura & Burnard, Pamela & Cooke, Carolyn & Davies, Richard & Gray, Donald & Trowsdale, Jo. (2017). BERA Research Commission Reviewing the potential and challenges of developing STEAM education through creative pedagogies for 21st learning: how can school curricula be broadened towards a more responsive, dynamic, and inclusive form of education? 10.13140/RG.2.2.22452.76161.

Cooper, S., Dann, W., & Pausch, R. (2000). Developing algorithmic thinking with alice. *ISECON*, 17, 506-539.

Cortina, T. J. (2015). Reaching a broader population of students through unplugged activities. *Communications of the ACM*, 58(3), 25–27.

Costantino, T. (2018). STEAM by another name: Transdisciplinary practice in art and design education. *Art Education Policy Review*, 119(2), 100-106. https://doi.org/10.1080/10632913.2017.1292973

Creswell, J. D. & Creswell, J. W. (2018). Research design: Qualitative, quantitative, and mixed methods approaches. *SAGE Publications Inc*.

Creswell, J. W., & Clark, V. L. P. (2011). Designing and conducting mixed methods research 2nd Edition. *SAGE Publications Inc*.

Cromley, J. G., Perez, T., Wills, T. W., Tanaka, J. C., Horvat, E. M., & Agbenyega, E. T. (2013). Changes in race and sex stereotype threat among diverse STEM students: Relation to grades and retention in the majors. *Contemporary Educational Psychology*, 38(2013), 247-258. https://doi.org/10.1016/j.cedpsych.2013.04.003

Daugherty, M. K. (2013). The prospect of an "a" in stem education. *Journal of STEM*

Education:Innovations and Research, 14(2), 10-15

Dickens, C. (1859). A tale of two cities. *Independently Published.*

Discourse. (2018). The merriam-webster dictionary (11th ed.). Merriam-Webster Mass Market.

Ehrlinger, J., Plant, E. A., Hartwig, M. K., Vossen, J. J., Columb, C. J., & Brewer, L. E. (2018). Do gender differences in perceived prorotypical computer scientist and engineers contribute to gender gaps in computer science and engineering? Sex Roles, 78(1), 40-51. https://doi.org/10.1007/s11199-017-0763-x

Fallon, G. (2016). An analysis of young students' thinking when completing basic coding tasks using scratch jnr. on the ipad. *Journal of Computer Assisted Learning*, 32(6), 576-593. https://doi.org/10.1111/jcal.12155

Fear, J. (2018). The knitting war spies of history. *Handwoven.*

Fitzallen, N., Reaburn, R., & Fan, S. (2014). The future of educational research: Perspectives from beginning researchers. *Rotterdam*.

Futschek, G. (2006). Algorithmic thinking: The key for understanding computer science. Informatics Education,159-168.

Gentry, L. (2014). Knit cowls. *Leisure Arts, Inc.*

Grover, S. & Pea. R (2013). Computational thinking in K-12: A review of the state of the field, *Educational Research*, 42(38), 38-43. https://doi.org/10.3102/0013189X12463051

Gulhan, F., & Sahin, F. (2018). Activity Implementation intended for STEAM (STEM + art) education: Mirror and light. *Journal of Inquiry Based Activities (JIBA)*, 8(2), 111-126. https://doaj.org/article/068d1a2d1aee4b3d9cba1c6dfa2d96f5

Gurbuz, H., Evlioglu, B., Erol, C. S., Gulsecen, H., & Gulsecen, S. (2016). "What's the weather like today?": A computer game to develop algorithmic thinking and problem solving skills of primary school pupils. *Education and Information Technologies*, 22, 1133-1147. https://doi.org/10.1007?s10639-016-9478-9

Guzdial, M. (2013). Exploring hypotheses about media computation. Proceedings of Ninth Annual *International ACM Conference on International Computing Education Research*, 19-26. https://dx.doi.org/10.1145/2493394.2493397

Hancock, D. R. & Algozzine, B. (2017). Doing case study research: A practical guide for beginning researchers (3rd ed.). Teachers College Press.

Harvard Graduate School of Education (2019). *Creative computing curriculum: Overview*. Creative Computing Curriculum | Overview. Retrieved May 2, 2020, from https://creativecomputing.gse.harvard.edu/guide/

Holland, D., & Cole, M. (1995). Between discourse and schema: Reformulating a cultural-historical approach to culture and mind. *Anthropology & Education Quarterly*, 26(4), 478–489.

Honigsfeld, A., & Dunn, R. (2003). High school male and female learning style similarities and differences in diverse nations. The Journal of Education Research, 96(4), 195-206, https://doi-org.ezproxy.uky.edu/10.1080/00220670309598809

Hui, A., Schatzki, T., & Shove, E. (2017). The nexus of practice: Connections, constellations, practitioners. Routledge.

Hunter-Doniger, T. and Sydow, L. (2016). A Journey from STEM to STEAM: A Middle School Case Study, *The Clearing House: A Journal of Educational Strategies*, Issues and Ideas, 89:4-5, 159-166, DOI: 10.1080/00098655.2016.1170461

Hurlburt, G. (2018). Thinking critically, about algorithmic thinking. IT Professional, 20(2), 5-10. htpps://doi.org/10.1109/MITP.2018.021921644

Hussain, S. (2008). Investigating models of computational learning theory. IEEE International Multitopic Conference, 418-422, https://doi.org/ 10.1109/INMIC.2008.4777774

Jawad, H. M., Trout, S., Abualkibash, M., & Xie, Y. (2018). Integrating art and animation in teaching computer programming for high school students. *2018 IEEE International Conference on Electro/Information Technology*, (0311-0317). https://ieeexplore-ieee-org.ezproxy.uky.edu/document/8500178

Jean Piaget Society. (2014). A Brief Biography of Jean Piaget. Retrieved from http://www.piaget.org/aboutPiaget.html

Kafai, Y. B. & Peppler, K. (2011). Youth technology, and diy: Developing participatory competencies in creative media production. Review of Research in Education, 35(1), 89-119.

Katai, Z. (2014). The challenge of promoting algorithmic thinking of both sciences- and humanities oriented learners. Journal of Computer Assisted Learning, 31(4), 287-299, https://doi.org/10.1111/jcal.12070

Korkmaz, O., Cakir, R., & Ozden, Y. (2015). A validity and reliability study of the computational thinking scales (CTS). *Computers in Human Behavior*, 72, 558-569. https://dergipark.org.tr/tr/pub/turkbilmat/issue/44381/385097

Kraemer, E. (2019). The purl code. *Ravelry*. https://www.ravelry.com/patterns/library/the-purl-code

Lee, J., & Junoh, J. (2019). Implementing unplugged coding activities in early childhood classrooms. *Early Childhood Education Journal*, 47(6), 709-716. https://doi.org/10.1007/s10643-019-00967-z

Liesaputra, V., Barmada, B., Ramirez-Prado, G., & Song, L. (2020). Future proofing kiwi kids through the use of digital technology. *SIGCSE* (20)

Looi, C., How, M., Longkai, W., Seow, P., & Liu, L. (2018). Analysis of linkages between unplugged activity and the development of computational thinking. *Computer Science Education*, 28(3), 255-279. https://doi.org/10.1080/08993408.2018.1533297

Massoud, L., Hallman, S., & Plaisent, M. (2018). Applying multidisciplinary teaching techniques to the computer programming/coding classroom. *Journal of IT and Economic Development*, 9(1), 38-47. http://web.a.ebscohost.com.ezproxy.uky.edu/ehost/detail/detail?vid=0&sid=09f8a3dc-f545-4189-bea1-4c6d58bc41c3%40sdc-v-sessmgr03&bdata=JnNpdGU9ZWhvc3QtbGl2ZSZzY29wZT1zaXRl#AN=133965734&db=bth

Maxwell, D. (2016). Coding is a necessary part of 21st century education. Retrieved from *The Educator*: https://www.theeducator.com/blog/coding-necessary-part-21st century-education/

Maxwell, J. A. (2013). Qualitative research design: An interactive approach. *SAGE Publications*

McVeigh-Murphy, A. (2020, January 8). *The one about abstraction in computational thinking*. equip. Retrieved April 1, 2022, from https://equip.learning.com/abstraction-computational-thinking/

Mediate. (2018). The merriam-webster dictionary (11th ed.). Merriam-Webster Mass Market.

Medina, C., & Wohlwend, K. E. (2014). Literacy, play, and globalization: Converging imaginaries in children's critical and cultural performances. Routledge.

Milroy, A., Holmes, A., & Wegener, M. J. (2015). Thinking in the arts - Science nexus labpunk curisoity, intra-actions and creativeness in a physics-art collaboration. Transformations *Journal of Media and Culture*, 26, 1-19, http://www.transformationsjournal.org/wp-content/uploads/2016/12/Milroy-Wegener-Holmes_Transformations26.pdf

Massoud, L., Hallman, S., & Plaisent, M. (2018). Applying multidisciplinary teaching techniques to the computer programming/coding classroom. Journal of IT and Economic Development, 9(1), 38-47. http://web.a.ebscohost.com.ezproxy.uky.edu/ehost/detail/detail?vid=0&sid=09f8a3dc-f545-4189-bea1-4c6d58bc41c3%40sdc-v-

National Science Foundation (2017) Science and Engineering Labor Force. https://ncses.nsf.gov/pubs/nsb20198/u-s-s-e-workforce-definition-size-and-growth

Palviainen, A. (2020). Video call as a nexus of practice in multilingual translocal families. *Zeitschrift für Interkulturellen Fremdsprachenunterricht*, 25(1), 85–108. http://tujournals.ulb.tu-darmstadt.de/index.php/zif

Papadakis, S., Kalogiannakis, M., Zaranis, N., & Orfanakis, V. (2016) Using scratch and app inventor for teaching introductory programming in secondary education: A case study. *International Journal of Technology Enhanced Learning,* 8(3/4), 217- 233.

Peppler, K., & Glosson, D. (2013). Stitching circuits: Learning about circuitry through e-textile materials. *Journal of Science Education and Technology*, 22(5), 751-763. https://doi.org/10.1007/s10956-012-9428-2

Peppler, K., & Wohlwend, K. (2018). Theorizing the nexus of STEAM practice. *Arts Education Policy Review*, 119(2), 88-99. https://doi.org/10.1080/10632913.2017.1316331

“Parallelize.” *Merriam-Webster.com Dictionary*, Merriam-Webster, https://www.merriam-webster.com/dictionary/parallelize. Accessed 18 Apr. 2022.

Petersen, G. A. J., & McClintock, M. (1942). A guide to codes and signals: International flag code, secret ciphers, weather signals, morse code, sign language, etc. *Whitman Publishing Company*.

Plymouth Yarn Design Studio (2017) Pendenza Sampler. https://www.ravelry.com/patterns/library/3220-sampler-shawl?buy=1

Resnick, M., Maloney, J., Monroy-Hernandez, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosebaum, E., Silver, J., Silverman, B., & Kafai, Y. (2009). Scratch: Programming for all, *Communications ACM*, 52(11), 60-67. https://doi.org/10.1145/1592761.1592779

Ricketts, R. (2018). Computational thinking for kindergarteners. Retrieved from https ://www.eduto pia.org/article/computational-thinking-kindergartners.

Roberts, S. (2019). Knitting is coding and yarn is programmable in this physics lab. *NY Times*. https://www.nytimes.com/2019/05/17/science/math-physics-knitting-matsumoto.html

Rubio, M. A., Romero-Zaliz, R., Manoso, C., & de Madrid, A. P. (2014). Closing the gender gap in an introductory programming course. *Computers & Education*, 82, 409-420. https://dx.doi.org/10.1-16/j.compedu.2014.12.003

Saez-Lopez, J., Roman-Gonzalez, M., & Vazquez-Cano, E. (2016). Visual programming languages integrated across the curriculum in elementary school: A two year case study using “scratch” in five schools. *Computers & Education*, 2016(97), 129-141. https://doi.org/10.1016/j.compedu.2016.03.003

Saldaña Johnny. (2021). *The coding manual for qualitative researchers*. SAGE Publishing.

Schlomer, R. (2019). Typographic knitting: From pixel to pattern. *Princeton Architectural Press.*

Scollon, R. (2001). Mediated discourse. Routledge.

Sepulveda-Diaz, C., Rojas, E. S., Simmonds, J., Gutierrez, F. J., Hitschfeld, N., Casanova, C., & Sotomayor, C. (2020). Lessons learned from introducing preteens in parent-led homeschooling to computational thinking. *SIGCSE* (20).

Starr, C. R. (2018). "I'm not a science nerd!": STEM stereotypes, identity, and motivation among undergraduate women. *Psychology of Women Quarterly*, 42(4), 489-503. https://doi-org.ezproxy.uky.edu/10.1177/0361684318793848

Sung, E. (2019). Fostering computational thinking. *Technology and Engineering Teacher*, 78(5), 8-13. http://search.ebscohost.com.ezproxy.uky.edu/login.aspx?direct=true&db=a9h&AN=134257017&site=ehost-live&scope=site

Taub, R., Armoni, M., & Ben-Ari, M. (2012). CS unplugged and middle-school students’ views, attitudes, and intentions regarding CS. *ACM Transactions on Computing Education* (TOCE), 12(2), 8. https://www.cs.auckland.ac.nz/courses/compsci747s2c/lectures/taub-cs-unplugged-toce-12.pdf

Taylor, C., Singaro, D., Clancy, M., Lee, C., Webb, K. C., & Porter, L. (2020). The practical details of building a cs concept inventory. *SIGSE* (20)

Thuneberg, H., Salmi, H., & Fenyvesi, K. (2017). Hands-on math and art exhibition promoting cience attitudes and educational plans. Education Research International 2017 (2017). file:///C:/Users/Emily/Documents/Dissertation/Articles/Art/Math%20and%20Art.pdf

Tobin D. D., Menon M., Menon M., Spatta B. C., Hodges E. V., Perry D. G. (2010). The intrapsychics of gender: a model of self-socialization. *Psychol Rev*. Apr;117(2):601-22. doi: 10.1037/a0018936. PMID: 20438239.

Tonbuloglu, B. & Tonbuloglu, I. (2019). The effect of unplugged coding activities on computational thinking skills of middle school students. *Informatics in Education*, 18(2), 403-426. https://doi.10.15388/infedu.2019.19

Toglia, T. (2013). Gender equity issues in CTE and STEM education. *TechDirections*, *72*, 14-18.

Tracy, S. J. (2013). Qualitative research methods: Collecting evidence, crafting analysis, communicating impact. *Blackwell Publishing.*

US Bureau of Employment in STEM Occupations : U.S. Bureau of Labor Statistics (bls.gov)

Varma, R. (2006). Making computer science minority-friendly: Computer science programs neglect diverse student needs. *Communications of the ACM*, 49(2), 129-134. https://doi-org.ezproxy.uky.edu/10.1145/1113034.1113041

Venkateswarlu, N. B. (2020). First book on bebras challenge and computational thinking: Hundi and English. *Amazon Services LLC*.

Venkateswarlu, N. B. (2019). *First Book on Bebras Challenge and Computational Thinking*. Independently Published.

Washington, Alicia & Grays, Shaefny & Dasmohapatra, Sudipta. (2016). The Computer Science Attitude and Identity Survey (CSAIS): A Novel Tool for Measuring the Impact of Ethnic Identity in Underrepresented Computer Science Students.

Weintrop, D. & Wilensky, U. (2019). Transitioning from introductory block-based and text-based environment to professional programming languages in high school computer science classrooms. *Computers and Education*, 142(2019), 1-17.https://doi.org/10.1016/j.compedu.2019.103646

Wohlwend, K. (2014). Mediated discourse analysis: Tracking discourse in action. Routledge.

Wohlwend, K. (2008). Kindergarten as a nexus of practice: A mediated discourse analysis of reading, writing, play and design in early literacy apprenticeship. Reading Research Quarterly, 43(4), 332-334. https://doi.org/g/10.1598/RRQ.43.4.1

Wohlwend, K. E., & Medina, C. L. (2012). Media as a nexus of practice:

Remaking identities in what not to wear. Discourse: Studied in the Cultural Politics of Education, 33(4), 545-560. http://dx.doi.org/10.1080/01596306.2012.692961

Yin, R. K. (2003). *Applications of case study research: Design and Methods* (Third). SAGE Publishing.

VITA

E. Dodson-Snowden

Education

2010-2011	Georgetown College, Graduate School of Education
	Ed.M in Initial Certification in Secondary Education
	Dr. Jonathan Thomas, advisor
2006-2010	Georgetown College
	B.S. in Environmental Science

Professional Experience

2012-present	Fayette Co. Public School, Lexington, KY