

ABSTRACT

WHITE, ISABELLA GRANSBURY. Investigating A New Pair Programming Collaboration Method For Block-Based Programming Environments. (Under the direction of Veronica Cateté).

A common collaboration method used in computer science (CS) education is called pair programming. During traditional pair programming, one student, the Driver, physically programs, while the other student, the Navigator, instructs the Driver on how to complete the program. Using student-centered collaborative methods, such as pair programming, with novice programmers in CS classes has shown increases in confidence, enjoyment, and positive attitudes among students. Despite the known benefits of this method, previous work shows that pair programming can be inequitable and not effective in K-12 CS education within certain student pairings. In previous work investigating pair programming in K-12 settings, researchers found that when students are paired with partners who wish to complete assignments faster than the other student, one partner will take over and will see the other student as having lower intellectual ability. In terms of effectiveness, researchers have shown that when students do not change roles correctly in intervals, the partner that spent most time as the Navigator can have lower learning gains compared to the partner who physically coded longer.

In an attempt to find a more effective collaboration method for novice computer science students, I present work that investigates student behavior and attitudes when using the Puzzle method, a new pair programming method. The Puzzle method is a variation of the traditional pair programming collaboration method which partitions the available blocks in a block-based programming environment equally to each student in a pair. Compared to traditional Driver-Navigator pair programming, the Puzzle method enables both students in a pair to act as Drivers by using concurrent editing of the same file. The partition of blocks attempts to distribute tasks evenly to students to provide a structure of how to communicate with each other to accomplish the end goal. More importantly, this distribution attempts to divide work equally so each student has equal input into the program. This work specifically investigates student behaviors when using the Puzzle method, effects of the Puzzle method on student attitudes, and how the Puzzle method can be improved for future use.

I modified a conversational framework from two previous works in pair programming to analyze student conversations while using the Puzzle method. The framework consisted of three elements: the distribution of the talk, the content of the talk, and the positioning of the students. I used an *a priori* codebook to analyze the content of student talk and created an additional tag "to self" that refers to students talking to themselves through challenges when completing an activity opposed to talking to their partner. The findings of this investigation

suggest that there tends to be a pattern of dialog interactions between pairs that are less collaborative versus more collaborative. Pairs that were less collaborative had a dynamic in which the more dominant participant said more statements, while the other participant said acknowledgments or asked questions. In these instances, the dominant participant also said a great amount of *to self* statements.

To determine the effects of the Puzzle on student attitudes, I distributed a pre and post survey to high school students during a week-long summer camp where they completed several programming activities using traditional pair programming and the Puzzle method. The study participants also completed a reflection questionnaire to gain more insight into their attitudes toward their partner. The attitudes constructs assessed in the pre and post survey include attitude toward collaboration, social support, confidence with computers, and programming confidence. The results of this study indicated that there were no statistically significant differences between student attitudes from the traditional pair programming group and the Puzzle. However, from the open ended questions of the reflection and post survey, we found more students preferred the Puzzle method than those who preferred the traditional method.

Finally, I applied the conversational framework to instances of a pair completing a program using the traditional pair programming method and the Puzzle method. I also conducted interviews with the participants in this study to determine their preferences between the two collaboration modes. We found that students use a broader amount of language to communicate with each other using the Puzzle method than the traditional method, and that the communication preferences of students should be considered when implementing collaborative programming practices in the classroom.

This dissertation makes several contributions. It is the first investigation of student behavior using a structured collaboration method in a block-based programming environment. I also implemented a conversational framework that combines two previous methodologies to give a more consistent method of identifying the student dialogue for future researchers. I showed that although there were no statistically significant differences between students' attitudes when using traditional pair programming and the Puzzle method, student preferences leaned towards the Puzzle method. Finally, I have conducted the first study comparing traditional pair programming with the Puzzle method that resulted in a deeper understanding of why students would prefer traditional pair programming over the Puzzle method and vice versa.

© Copyright 2024 by Isabella Gransbury White

All Rights Reserved

Investigating A New Pair Programming Collaboration Method For Block-Based Programming
Environments

by
Isabella Gransbury White

A dissertation submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Computer Science

Raleigh, North Carolina
2024

APPROVED BY:

Tamecia Jones

Sarah Heckman

Sandeep Kuttal

Veronica Cateté
Chair of Advisory Committee

DEDICATION

To my parents, grandfather, and husband.

BIOGRAPHY

The author has lived in many different places and been on many adventures. She is most passionate about improving computer science education and increasing access to computing education for underrepresented peoples in computer science. She would be more than willing to share her story if you ask, but it is a bit of a roller coaster.

ACKNOWLEDGEMENTS

I would like to thank my advisor, Dr. Veronica Cateté, my mentor, Dr. Sarah Heckman, and other members of my committee, Dr. Sandeep Kuttal and Dr. Tamecia Jones for their time and support throughout this work.

I would also like to thank my parents, Amanda and Zechariah Gransbury and my grandparents John and Sue Powell for their love, support, and encouragement not only through my graduate school journey, but the entirety of my academic career. Additionally, I would like to thank my husband for his love and support through my journey in graduate school and always being willing to let me talk to him about my research.

I would also like to thank The Engineering Place at NC State for allowing me to conduct my studies in their summer camps and for giving me the opportunity to help so many kids interested in computer science.

I would like to thank members of the Game2Learn lab and CEREAL lab for their support and proof reading my work. I would like to include a special thanks to Lauren Alvarez and Madison Thomas for their support with my studies. Additionally, I would like to thank the undergraduate researchers who assisted with my research studies: Monica Jin, Brooke Wu, Oro Kang, and Emily Root, Sophie Goeuriot, and Kyla Griffin.

Last but not least, I would like to thank my high school programming teacher, Sharon McPherson, for encouraging me to pursue a career in computer science.

TABLE OF CONTENTS

List of Tables	viii
List of Figures	ix
Chapter 1 INTRODUCTION	1
1.1 Motivation	1
1.2 Research Questions and Hypotheses	2
1.3 Contributions	3
Chapter 2 Block Based Programming and The Puzzle Method	5
2.1 Scratch	6
2.1.1 ScratchJr	7
2.2 Unreal Engine: Blueprints	8
2.3 Blockly	8
2.4 Snap!	9
2.5 NetsBlox	10
2.6 The Puzzle Method	11
Chapter 3 Literature Review	15
3.1 Collaborative Learning	15
3.2 Pair Programming in Undergraduate CS Education	16
3.3 Pair Programming in K-12 Computer Science Education	17
3.3.1 Elementary School	17
3.3.2 Middle School	18
3.3.3 High School	18
3.4 Other investigations of pair programming	19
3.4.1 Pairings During Pair Programming	19
3.4.2 Affects of Pair Programming on Girls	19
3.4.3 Variations of Pair Programming	20
3.4.4 Pair Programming and Equity	20
3.5 Discussion	20
3.6 Method Used In Dissertation	21
Chapter 4 A Conversational Analysis of Secondary Students Using the Puzzle Pair Programming Method.	23
4.1 Introduction	23
4.2 Research Questions	25
4.3 Background	25
4.3.1 Variations of Pair Programming	25
4.3.2 Pairings During Pair Programming	26
4.3.3 Effects of Pair Programming on Girls	26
4.3.4 Pair Programming and Equity	26
4.4 Research Methods	27

4.4.1	Study Context	27
4.4.2	Participants	27
4.5	Data Analysis	28
4.5.1	RQ1a: Distribution of Student Talk	28
4.5.2	RQ1b: Content of Student Talk	29
4.5.3	RQ1c: Student Positioning	30
4.6	Results	30
4.6.1	Pair 1: M & M	30
4.6.2	Pair 2: F & F	32
4.6.3	Pair 3: F & M	33
4.6.4	Pair 4: M & M	34
4.7	Discussion	36
4.8	Conclusion	37
Chapter 5	Student Attitudes and Perceptions of the Puzzle Method versus Traditional Pair Programming	39
5.1	Introduction	39
5.2	Research Questions	40
5.3	Background	40
5.4	Research Methods	41
5.4.1	Study Context	41
5.4.2	Participants	42
5.5	Data Analysis	43
5.5.1	Pre and Post Survey	43
5.5.2	End Reflection	43
5.5.3	Thematic Analysis	44
5.6	Results	44
5.6.1	RQ2a: Student attitudes	45
5.6.2	RQ2b: Student Preferences	47
5.6.3	RQ2c: Student Perceptions	49
5.7	Discussion	52
5.8	Conclusion	53
Chapter 6	Conversational Analysis and Attitudes of Students Using the Puzzle Method versus Traditional Pair Programming	56
6.1	Introduction	56
6.2	Research Questions	57
6.3	Background	57
6.4	Research Methods	58
6.5	Data Analysis	60
6.6	Conversational Analysis	60
6.6.1	Thematic Analysis	60
6.7	Results	61
6.7.1	RQ3a: Distribution of Talk	62
6.7.2	RQ3b: Content of Talk	63

6.7.3	RQ3c: Student Positioning	64
6.7.4	RQ3d: Student Preferences	64
6.8	Discussion	68
6.9	Conclusions	69
Chapter 7	CONCLUSIONS	73
7.1	Review	73
7.2	Research Questions	73
7.3	Hypotheses	74
7.4	Summary of Results	74
7.5	Contributions	80
7.6	Future Work	81
References	82
APPENDICES	89
Appendix A	Survey	90
Appendix B	Interview Protocol	92
Appendix C	Example Programming Activity	93
Appendix D	Accessing the Puzzle Method in NetsBlox	102

LIST OF TABLES

Table 1.1	The research questions, associated hypotheses, and their locations in this work.	4
Table 4.1	Self-identified genders of participants within each pair.	27
Table 4.2	<i>A priori</i> tags used to analyze RQ1b.	29
Table 5.1	Means of participant responses to attitudes toward collaboration and social support from the pre survey.	42
Table 5.2	Mean values of student responses from the pre and post survey from both groups about attitudes toward collaboration.	45
Table 5.3	Mean values of student responses from the pre and post survey from both groups about participants' social support with technology.	46
Table 5.4	Mean values of student responses from the pre and post survey from both groups about participants' confidence with computers.	46
Table 5.5	Mean values of student responses from the pre and post survey from both groups about participants interest in programming.	47
Table 5.6	Count and student responses from Q# on the post survey which asked participants "Did you find the pair programming (Puzzle or traditional) method to be helpful when completing the programming activities?" . . .	47
Table 5.7	Themes found from the open responses in the post survey relating to traditional and Puzzle pair programming.	48
Table 5.8	Mean values of student responses from the post survey from both groups about participants views of their partners.	50
Table 5.9	Themes found from the open responses in the reflection relating to participant's and their partners collaboration behaviors.	55
Table 6.1	<i>A priori</i> tags used to analyze RQ3b.	61
Table A.1	Questions asked in the pre and post survey, the constructs they measure, and their respective answer scales.	91
Table B.1	The questions asked in the participant interviews in Chapter 6	92

LIST OF FIGURES

Figure 2.1	Scratch's programming environment with element annotation.	6
Figure 2.2	Example screen shot of the ScratchJr programming environment.	8
Figure 2.3	Example screen shot of the Blueprints programming environment.	9
Figure 2.4	Example screen shot of the Blockly programming environment.	10
Figure 2.5	Example screen shot of the Snap! programming environment.	11
Figure 2.6	Available block pallets in the NetsBlox programming environment.	12
Figure 2.7	Example partition of blocks between two students while using the Puzzle method.	13
Figure 2.8	Example screen of Student A.	14
Figure 2.9	Example screen of Student B.	14
Figure 4.1	Percentage of each type of tag expressed during the activity for pair 1. . .	31
Figure 4.2	Percentage of each type of tag expressed during the activity for pair 2. . .	32
Figure 4.3	Percentage of each type of tag expressed during the activity for pair 3. . .	33
Figure 4.4	Percentage of each type of tag expressed during the activity for pair 4. . .	35
Figure 6.1	Screenshot of Partner 1's available block pallets.	59
Figure 6.2	Screenshot of Partner 2's available block pallets.	59
Figure 6.3	Distribution of talk between Partner 1 and Partner 2 while using traditional pair programming.	62
Figure 6.4	Distribution of talk between Partner 1 and Partner 2 while using Puzzle pair programming.	62
Figure 6.5	Percentage of tags during the activity completed with the traditional pair programming method	71
Figure 6.6	Count of tags for each participant during the activity completed with the traditional pair programming method	71
Figure 6.7	Percentage of tags during the activity completed with the Puzzle pair programming method	72
Figure 6.8	Count of tags for each participant during the activity completed with the Puzzle pair programming method	72

CHAPTER

1

INTRODUCTION

1.1 Motivation

In this work, I investigate the effects of a new method of pair programming, called Puzzle, in an attempt to establish if using structured collaboration creates an equitable and effective collaboration method for block-based programming environments (BBP). During pair programming, one student acts as the "Driver" and physically codes, while the other student gives directions to complete the program, the "Navigator" (Williams et al. 2000). When using Puzzle, students act as Drivers and Navigators based on the block pallets they have access to in the BBP environment. The Puzzle method will be discussed in greater detail in Chapter 2

Using student-centered collaborative methods, such as pair programming, with novice programmers in computer science classes has shown increases in confidence (Williams et al. 2000), enjoyment (Howard et al. 2006), and positive attitudes among students (Lytle et al. 2020). Despite the known benefits of this method, previous work shows that pair programming can be inequitable and not effective in certain situations (Lewis and Shah 2015; Shah et al. 2014; Tsan et al. 2020). This can cause one student in each pairing to become dominant which can lead to them portraying themselves as the more intelligent partner (Lewis and Shah 2015). This portrayal can have negative affects for the other partner such as negative attitudes toward collaboration and lack of confidence.

When focusing on equitable learning environments, researchers found that when students are paired with partners who wish to complete assignments faster than the other student, one partner will take over and will see the other student as having lower intellectual ability (Lewis and Shah 2015). In terms of effectiveness, researchers have shown that when students do not change roles correctly at intervals, the partner who spends the majority of the time giving instructions can have lower learning gains compared to the partner who physically coded longer (Rodríguez et al. 2017). Another researcher found that the use of two computers, rather than sharing one, leads partners to feel disconnected from their partner (Tsan et al. 2020). In an attempt to find a more effective collaboration method for novice computer science students, I present work that investigates student behavior when using the Puzzle method, effects of the Puzzle method on student attitudes, and how the Puzzle method can be improved for future use.

1.2 Research Questions and Hypotheses

Due to the inconsistencies of results in previous works investigating pair programming in K-12 settings, I have decided to focus on determining if block availability, structured collaboration encourages students to have more collaborative behaviors and more positive attitudes towards collaboration and computer science. The structure of the Puzzle method is derived by the availability of block palettes between two students in a pair, compared to the traditional pair programming method that uses the student roles, driver and navigator, to implement structure. The research questions and hypotheses I investigate in this work are listed in Table 1.1. The purpose of this work is to give more insight to the K-12 computing education research community on how students communicate when collaborating with structured roles and what the effects of that collaboration are.

I begin this work by analyzing the behaviors of four pairs of students while they complete a programming activity while using the Puzzle method (Chapter 4). I then compare the attitudes of students after completing a several programming activities while using the traditional pair programming method, and the Puzzle method (Chapter 5). In Chapter 6, I compare student behaviors when using the Puzzle method and the traditional pair programming method. I also conduct a participatory design study to gain insight into what students may need from the block-based programming interface to make collaborating easier (Chapter 6). My contributions are briefly listed in the next section and are discussed further in Chapter 7. Chapter 2 presents an introduction to block-based programming environments and the puzzle method. I also present a literature of previous work concerning pair programming in Chapter 3. Those familiar with my work may wish to start in Chapter 4.

1.3 Contributions

1. I conducted the first study investigating student behavior while using a collaboration method that uses block palettes in a block-based programming environment to divide tasks among pairs when pair programming. From this study, we were able to gain insight into collaborative behavior patterns, such as question-answer, that may indicate more equitable collaboration between partners.
2. I conducted the first study to investigate student attitudes, collaboration preferences, and perceptions of their partner after using traditional and Puzzle pair programming. From this study, we were able to show that the Puzzle method may have a positive affect on student attitudes toward collaboration and computing confidence. We were also able to show that students felt they communicated more using Puzzle although, some students did not enjoy it.
3. I conducted the first study to compare student behavior when using the traditional Driver-Navigator method between behavior when using the Puzzle method. From this study, we were able to gain insight into how the same students collaborate in each method. We found that when students collaborate using the Puzzle method, they use a wider range of language to communicate with each other, such as asking questions or sharing meta-comments, than when using traditional pair programming.
4. I modified a hybrid conversational analysis framework that combined the analysis of student talk distribution, content of talk, and student positioning with an established *a priori* codebook used for discourse analysis of pairs. This allows the computing education research community to have a consistent framework to evaluate student behavior in pair programming studies and compare results to other work.
5. I provided the computing education research community with suggestions of how to better investigate pair programming with secondary students and how to use block palettes in a block-based programming environment to divide tasks among pairs when pair programming. I also provided the K-12 computer science education community with implications of using Puzzle pair programming in the classroom which include, using Puzzle as a learning tool before pair programming and to consider student's preferences of communication when creating pairs for pair programming.

Table 1.1: The research questions, associated hypotheses, and their locations in this work.

Research Questions and Hypotheses	Chapter
[RQ1] How do students collaborate when using the Puzzle method?	
[H1] Participants will have a more even distribution of talk because tasks will be evenly distributed to each participant. [H2] When using the Puzzle method participants will be more engaged with each other when communicating verbally through questions and giving responses. [H3] Very little student positioning will occur between students because one student will be unable to "take over" the program.	Chapter 4
[RQ2] How are student perceptions about collaboration and computer science affected by when using the Puzzle method compared to traditional pair programming?	
[H4] Participants who complete activities using the Puzzle method will have statistically significant changes in their attitudes towards computer science. [H5] Participants will enjoy the Puzzle method more than traditional pair programming because it gives their roles during collaboration more structure and easier communication. [H6] Participants who enjoyed working with their partner will have positive attitudes towards the pair programming method they used.	Chapter 5
[RQ3] How do student behaviors differ when students use the traditional pair programming method compared to the Puzzle method?	
[H7] When using the Puzzle method, participants will have a more equal distribution of talk than when using the traditional method. [H8] When using the Puzzle method, participants will use similar types of language to talk to each other (such as questions and answers) than when using the traditional method. [H9] When using the Puzzle method, less student positioning will occur than when using the traditional method. [H10] Students will prefer the Puzzle method over the traditional method because it encourages engagement with their partner and is an overall more fun and enjoyable way to collaborate.	Chapter 6

CHAPTER

2

BLOCK BASED PROGRAMMING AND THE PUZZLE METHOD

Block-based programming (BBP) languages are educational tools to teach novice students computer science concepts. A benefit of these tools is the ability to have a visual representation of actions being performed in code. Another benefit is the allowance for students to focus on the computing concepts they are learning opposed to the syntax of the language, which can lead to frustration and lack of interest in computer science in the future (Weintrop 2019). Previous work has shown that the approach of BBP languages makes it accessible and less intimidating for beginners (Perera et al. 2021). Its user-friendly interface is particularly effective in engaging young students and those new to programming. There have also been studies that show BBP languages can increase students' interest and engagement with technology, serving as a stepping stone to more advanced computing topics and programming languages (Lin and Weintrop 2021). These tools are frequently recognized for their ability to introduce and reinforce computational thinking skills among students. Researchers have found that they can help learners understand abstraction, algorithms, and data structures in a more engaging way than traditional programming languages (Almanza Cortés et al. 2019; Çakıroğlu et al. 2021).

BBP environments also allow students to create interactive stories, games, and animations, which can lead to a deeper understanding of the subjects by encouraging creativity and

problem-solving. Educators have used BBP as a pedagogical tool to teach not only coding but also mathematics, science, digital storytelling, and other subjects (Smith et al. 2020; Weintrop and Wilensky 2017; Gleasman and Kim 2020). Its versatility allows for interdisciplinary approaches to education. Previous work has focused on how BBP environments can be integrated into the curriculum and the importance of training teachers to use these environments effectively in the classroom (Vasconcelos and Kim 2020; Dağ 2019). In this chapter, I will describe the history of BBP environments, the differences between various environments that have been developed for educational purposes, and how the Puzzle method was derived from these tools.

2.1 Scratch

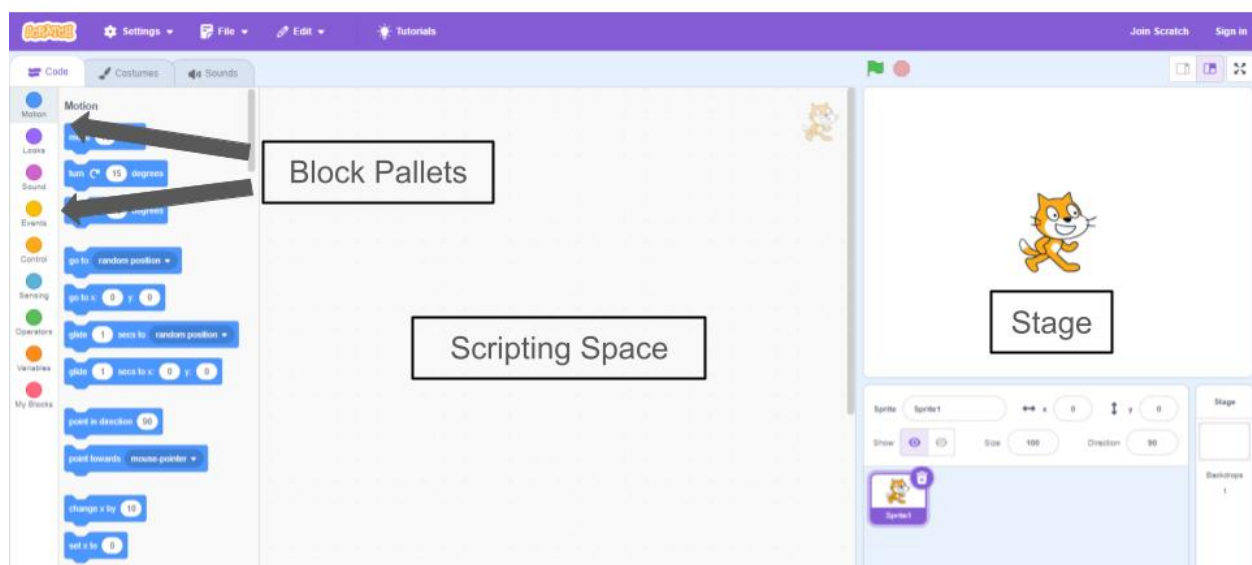


Figure 2.1: Scratch's programming environment with element annotation.

Modern BBPs originated in MIT's Media lab by Mitchel Resnick and Yasmin Kafai Maloney et al. (2004) (see Figure 2.1). Resnick and Kafai et al. created a programming environment called *Scratch* where students can click and drag blocks to create programs instead of writing lines of code. This presented the computer science education community with the ability

to represent code and its output visually. This ability opened the doors for a wide range of learners to experience computer science in a new and engaging way that had never been taught before. Scratch originated from the environment *etoys* (Kay et al. 1997), which was developed from the object-oriented and reflective programming language *Squeak* (Ingalls et al. 1997). Although *etoys* and *Squeak* were developed and used before Scratch, we begin with the modern BBP environment because it has a similar environment and functionality to other BBP environments that play a role in the development of the Puzzle method. The Scratch environment consists of several elements which include: block pallets, the scripting space, and the stage.

Block pallets hold the blocks available in the environment and are displayed similarly to website menus. There are eight block palettes: motion, looks, sound, pen, control, sensing, operators, and variables. Each of these pallets contains blocks that correspond to a computing concept, such as control, or the output of the program, such as pen. When a student wants to use a block in their program, they click and drag the block to the scripting space. The scripting space is a large space typically in the middle of the environment where students assemble their programs. Output from student programs created in the scripting space is shown on the stage. The stage is a display window to the right of the screen where output of programs can be seen.

2.1.1 ScratchJr

In 2016, ScratchJr was released in 2016 as a simplified version of the original Scratch programming environment, designed specifically for younger children (ages 5-7) (see Figure 2.2). There are several differences between the original Scratch environment and ScratchJr including interface complexity, programming blocks, and functionalities.

ScratchJr has a simplified interface with fewer options and commands to make it more accessible for young children who may not have developed reading and math skills yet. Additionally, the blocks are more basic and use symbols instead of text to represent their functionality, making it easier for non-readers to understand and use them. Finally, Scratch has more advanced functionalities such as custom variables, lists, and a wider variety of control structures, whereas ScratchJr sticks to the basics, focusing on simple motion, looks, sound, and control blocks. In summary, ScratchJr is designed with the cognitive and motor skills of young children in mind, providing an age-appropriate introduction to programming concepts. As children grow older and more capable, they can transition to Scratch to explore more complex programming possibilities.



Figure 2.2: Example screen shot of the ScratchJr programming environment.

2.2 Unreal Engine: Blueprints

Unreal Engine's Blueprints is a BBP environment specifically designed for creating game play mechanics and interactive elements in the Unreal Engine game development platform (see Figure 2.3). The main differences between Blueprints and Scratch are the target audience for the tool, functionality, and interface. While Scratch and ScratchJr are designed for beginners and younger users, Blueprints is aimed at more advanced developers and users who have a firm understanding of game development concepts. Blueprints offers a wide range of advanced features and functionalities specifically tailored for game development, including complex logic, physics simulations, and AI scripting. ScratchJr and Scratch have a simpler and beginner-friendly user interface, with colorful blocks representing different programming elements. In contrast, Blueprints has a more professional and complex interface, with a focus on providing advanced tools and options for game developers.

2.3 Blockly

Blockly is a BBP environment developed by Google (see Figure 2.4). It is designed to make the principles of coding accessible and understandable, particularly for educational purposes and for users who are new to programming. Similar to Scratch, Blockly features a drag-and-

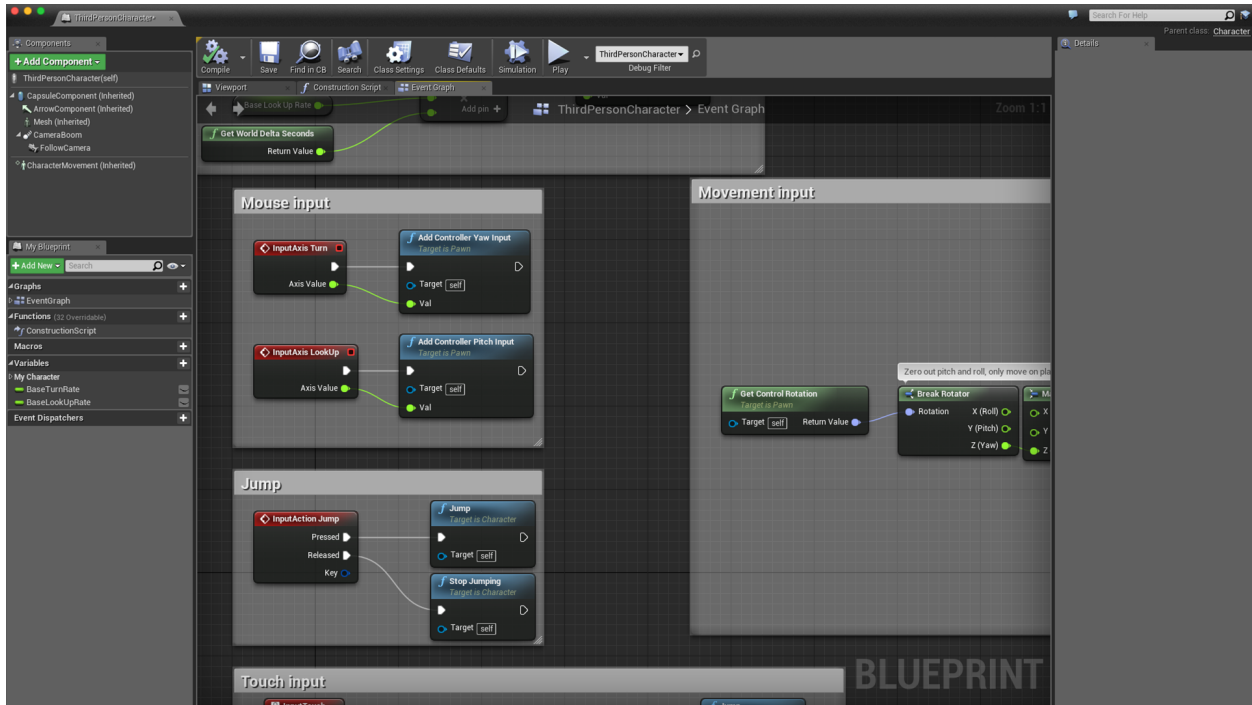


Figure 2.3: Example screen shot of the Blueprints programming environment.

drop interface where users can select interlocking blocks representing different programming statements and structures. The blocks are organized into pallets similar to Scratch such as controls, logic, loops, math, and variables, among others. Users can easily browse through these categories to find the blocks they need for their programs.

One of the key features of Blockly is its ability to generate syntactically correct code in several programming languages, such as JavaScript, Python, PHP, Lua, and Dart. This feature helps users understand how block-based code translates into traditional programming languages and can facilitate the transition from visual programming to text-based coding. Users can export the code they have created with blocks into a textual code format, assisting in the learning of traditional text-based coding. The design of Blockly is such that it can be embedded into web pages and online educational tools, as well as be tailored for specific use cases, making it a flexible option for educators and developers who wish to incorporate BBP into their platforms.

2.4 Snap!

Snap! is known to be an extension of Scratch, with some additional capabilities and differences. One difference is the extended block pallet, which allows for more complex operations and data structures, such as lists. Lists are a type of data type that can store lists or other variables, making

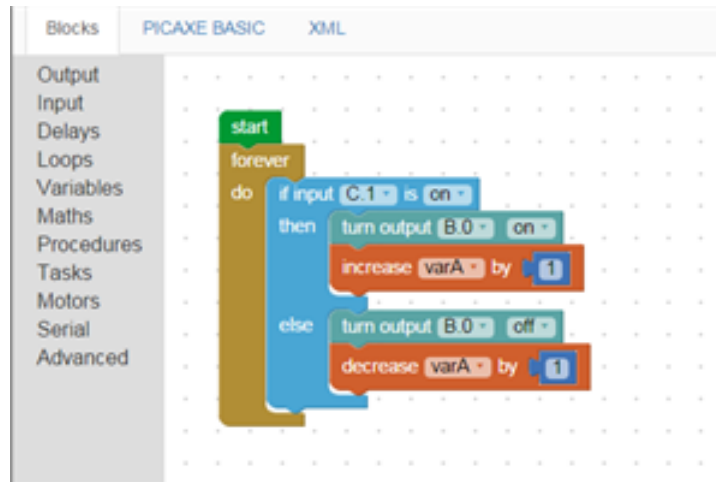


Figure 2.4: Example screen shot of the Blockly programming environment.

them a convenient way to store and access multiple variables at once. Another difference is the creation of custom blocks more freely in Snap!, with inputs of various data types and varying amounts. These custom blocks can be used as inputs in other blocks, which allows for high-order functions. A significant difference is the act of sprite inheritance, which allows one sprite to inherit behaviors and scripts from another similar to object-oriented programming.

These differences increase the complicity of Snap!, making it suitable for learners with more programming experience. Snap! is often used in more formal educational settings and in computer science classrooms to teach introductory concepts. Overall, while Scratch and Snap! share the core BBP approach and an emphasis on education and creativity, Snap! offers a more advanced set of features aimed at deeper computer science learning and exploration. It allows users to progress to concepts typically taught in text-based programming languages while still using a visual programming environment.

2.5 NetsBlox

The newest BBP environment is called NetsBlox created by Akos Ledeski and Brian Broll at Vanderbilt University (Broll et al. 2016). This environment is an extension of Snap! and includes two new block palettes, Services and Custom (see Figure 2.6). These additional block pallets allow students to connect to several Application Programming Interfaces (APIs) which allows students to create interdisciplinary projects and have access to online datasets (Broll et al. 2017). Examples of these datasets include, but are not limited to, climate and weather data, New York Times articles, and song lyrics.

This BBP environment was created to combine the idea of API's and BBP into one envi-

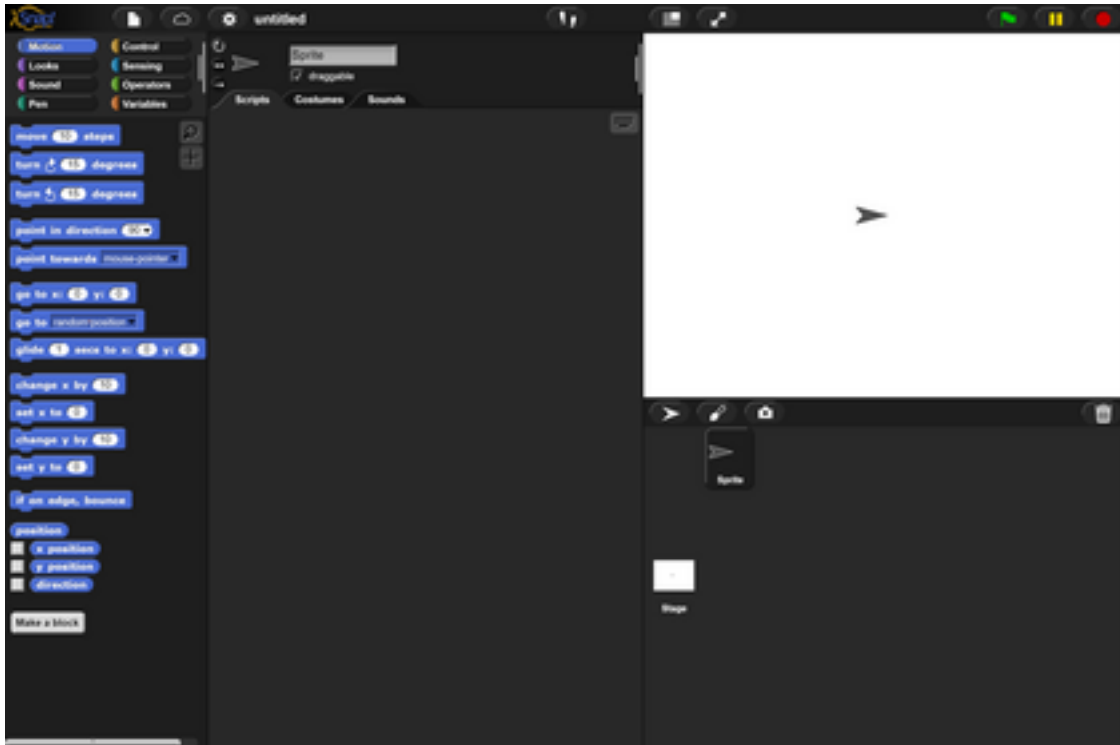


Figure 2.5: Example screen shot of the Snap! programming environment.

ronment for use in a curriculum for high school students, Computer Science Frontiers. The purpose of this course is to introduce advanced computer science (CS) topics to students with varying previous programming experience in attempts to garner their interest in CS, since early career decisions are made at this age. With the use of API's in conjunction of a BBP language, students can create a wide range of projects on topics they are interested in. This can spark interest in CS in students who are usually not drawn to the subject (Broll et al. 2021). NetsBlox also contains a collaboration feature, similar to Google Docs, that allows students to work on a project simultaneously on different machines (Brady et al. 2022). This concurrent editing feature is crucial in the implementation of the Puzzle method.

2.6 The Puzzle Method

The Puzzle method is a variation of the pair programming collaboration method that partitions the block pallets available to each student in a BBP environment (Lytle et al. 2020). The partition of blocks attempts to distribute tasks evenly to students to provide a structure of how to communicate with each other to accomplish the end goal. Compared to traditional Driver-Navigator pair programming, the Puzzle method enables both students in a pair to act as



Figure 2.6: Available block palettes in the NetsBlox programming environment.

Drivers by using concurrent editing of the same file.

In Figure 2.7 we can see the partition of blocks in a programming script completed by two students using the Puzzle method. Student A has access to control (yellow) and variable (orange) blocks, while student B has access to sensing (light blue), motion (dark blue), and operation (green) blocks. In the programming environment, students can only see blocks they have access to. In Figure 2.8 and Figure 2.9 you can see the views of Student A and Student B from the example in 2.7.

More importantly, this distribution attempts to divide work equally so each student has equal input into the program. For example, in Figure 2.7 we can see the partition of blocks in a programming script completed by two students using the Puzzle method. In total, Student A contributed a total of 14 blocks and Student B contributed a total of 13 blocks. The partition of blocks between students may differ based on the desired solution to an activity. The main purpose of the partition is for each student to have equal contributions of their assigned programming concepts that are associated with each block. The partition of blocks also attempts to address the student detachment found when students used two separate computers (Tsan et al. 2020). Since students can only see blocks assigned to them, a necessity has been created to discuss with their partner which blocks are needed to complete the activity. This eliminates the possibility of the "divide and conquer" method and increases the need for collaborative

Completed Code	Student A's Blocks
<pre> when clicked set Lives to 3 set Score to 0 forever if Touching Car change Lives by -1 go to x: 0 y: -150 if Lives < 1 Game Over if touching LilyPad? change Score by 10 go to x: 0 y: -150 if key up arrow pressed? change y by 5 if key down arrow pressed? change y by -5 if key left arrow pressed? change x by -5 if key right arrow pressed? change x by 5 </pre>	<pre> when clicked forever if Lives change by 1 set to 0 Game Over </pre>
	Student B's Blocks
	<pre> touching ? Touching Car key space pressed? change x by 10 change y by 10 go to x: 0 y: 0 < > </pre>

Figure 2.7: Example partition of blocks between two students while using the Puzzle method.

talk in order to complete an activity.

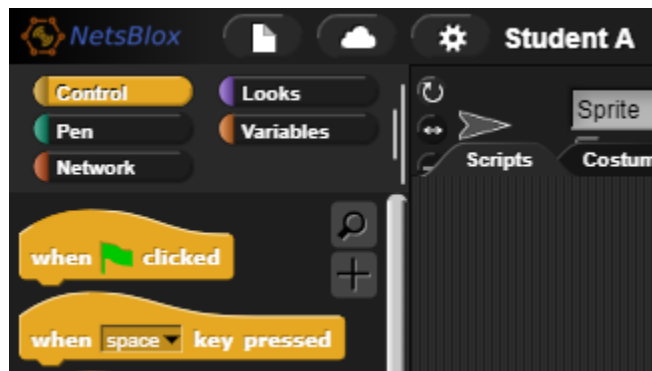


Figure 2.8: Example screen of Student A.

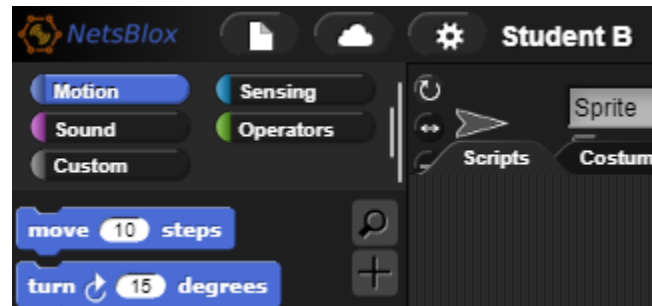


Figure 2.9: Example screen of Student B.

CHAPTER

3

LITERATURE REVIEW

In this chapter, I will provide an overview of previous work conducted to investigate pair programming in various educational settings. The work here provides a foundation for the work proposed in Chapter 4, Chapter 5, and Chapter 6. Each chapter mentioned previously contains a motivation section; however, this chapter should be used as background knowledge for this entire body of work.

3.1 Collaborative Learning

The idea of collaborative learning stems from Lev Vygotsky's concept *The Zone of Proximal Development* (Vygotsky et al. 2011). Vygotsky defined this concept as "the distance between the actual developmental level as determined by independent problem solving and the level of potential development as determined through problem solving under adult guidance or in collaboration with more capable peers." Collaborative learning is then the act of student development when problem solving with their peers. This collaboration can happen face-to-face and over online mediums such as online discussion boards. Pair programming has many benefits including increased engagement, more effective learning, and more positive attitudes (Tsang et al. 2023).

In computer science (CS), examples of collaborative learning include group projects, peer-

to-peer instruction, and pair programming; With pair programming being one of the most common methods of collaboration. Pair programming was spear headed by Dr. Laurie Williams, a distinguished professor of computer science. In her description of pair programming, Williams states "One of the programmers, the driver, has control of the keyboard/mouse and actively implements the program. The other programmer, the observer, continuously observes the work of the driver to identify tactical (syntactic, spelling, etc.) defects, and also thinks strategically about the direction of the work (Williams 2001)." In this work, the roles of the Driver and Navigator are relevant to the investigations of how Driver roles will effect collaborative behavior when using the Puzzle method.

3.2 Pair Programming in Undergraduate CS Education

The majority of pair programming studies have been conducted with undergraduate computer science students (Williams et al. 2000). McDowell et al. investigated the effects of pair programming on performance with 600 undergraduate students in an introductory programming course (McDowell et al. 2002). The results indicated that the students who programmed in pairs performed better than those who programmed individually and established that collaboration is an effective pedagogy to teach introductory computer science concepts (McDowell et al. 2002). In another study investigating pair programming in a second-year undergraduate software development course, Mendes et al. also found that pair programming is an effective learning technique (Mendes et al. 2005). Further research on pair programming found that it is the most effective when partners have equal conversational contributions and resolve the challenges they face (Rodríguez et al. 2017). These results not only established that pair programming can help students with various levels of programming knowledge learn new programming skills, but also established suggestions when pair programming is the most beneficial to learning.

After establishing pair programming as a beneficial pedagogy in post secondary computer science education, researchers began to investigate the affect it has on students. In a study investigating the effects of pair programming on students with low Scholastic Aptitude Test scores, Braught et al. found that students who pair program have higher confidence levels (Braught et al. 2011). When investigating the effects of pair programming, McDowell et al. found that it also increases student confidence, computer science proficiency, and improved program quality. These authors also hypothesize that pair programming can be used to increase women's retention in computer science (McDowell et al. 2006).

Previous work examining the effects of pair programming on women helps support this claim (Werner et al. 2004; Maguire et al. 2014). In Werner et al., the authors recognize the effects

of pair programming for all students, but emphasize the effect that it has on women. The results of their study explain how pair programming can remove the competitive nature of computer science which shows how programming is not a "socially isolating" activity (Werner et al. 2004). Maguire et al. compared the final exam grades of students who programmed in pairs at their university. They found no significant increase for male final exam scores, but found a significant increase for females (Maguire et al. 2014). Despite the context of the proposal primarily focusing on secondary students, the previous work done in post-secondary education provides an overview of the benefits of pair programming and perhaps why it was introduced to K-12 computer science education settings.

3.3 Pair Programming in K-12 Computer Science Education

Major focuses in pair programming investigations with primary and secondary students includes effectiveness, how to pair students, and the effect on underrepresented groups in computing, such as women. In this section, I will describe previous work investigating pair programming in K-12 educational settings that provides the basis motivation for this work.

3.3.1 Elementary School

In K-12 computer science education, pair programming can be implemented with students as early as elementary school. The more recent work with this age group is slightly contradictory but may have been affected by factors that took place outside of schools.

In Iskrenovic-Momcilovic, the authors initiated pair programming using the block-based programming (BBP) environment Scratch. They found that there are also positive effects when using pair programming in a BBP (Iskrenovic-Momcilovic 2019). However, they did not find a significant difference between the performance of boys and girls. In another study investigating student attitudes when using pair programming in Scratch virtually, researchers found that students enjoyed pair programming, but there were no clear benefits to the pedagogy of their learning (Bodaker and Rosenberg-Kima 2022). The authors of this work note that the COVID-19 pandemic may have caused these results, as students may not have socially engaged with other students as often as they would in person. This would create some disconnect between the collaboration and communication skills students are assumed to have in order for pair programming to be successful, especially with elementary students.

A common occurrence in pair programming studies with elementary students is using more than one computer. In Wei et al., the authors investigated the self-efficacy and computational thinking skills of elementary students using "partial pair programming". This is described as

letting students collaborate while completing a task as individual students (Wei et al. 2021). This was done to eliminate the monitoring the teacher would have to conduct to ensure that the students switched roles at indicated intervals. Wei et al. found that the partial pair programming approach increases student self-efficacy and would be a beneficial collaboration method to use with students. They also found that the teachers enjoyed using the method in their classroom. In a similar study, (Tsan et al. 2020) compared students' preferences when pair programming with one computer versus two computers. The results indicated that there were "trade-offs" with both methods (more details are discussed in Section 3.4.3. Another study using a two computer pair programming method found that pairs that were given feedback during the collaboration process tend to have more collaborative behaviors (Zakaria et al. 2022).

3.3.2 Middle School

In pair programming studies with middle school students, the results can also be contradictory. In (Werner and Denning 2009), the authors performed an observational study of middle schools using pair programming (Werner et al. 2009). The results indicated that there are concrete actions that undermine effective problem solving, such as students positioning themselves as more intelligent than each other (Werner et al. 2009). Another study investigating pair programming with middle school students found that it is essential that students change roles to ensure that pair programming is beneficial to students (Denner et al. 2021). In a study investigating student attitudes towards collaboration, researchers found pairs where one student had a more positive attitude than their partner, their knowledge decreased (Denner et al. 2014).

3.3.3 High School

Despite the large amount of literature of pair programming studies involving elementary and middle school students, there are only several studies investigating high school students. Papadakis found positive results in a study conducted investigating the effects of pair programming on first-year secondary students (Papadakis 2018). The results of the study indicated students' attitudes toward computing improved and that they also had improved comprehension of introductory programming concepts (Papadakis 2018). In another study investigating the effects of pair programming on secondary students, Gray et al. found not only positive results for students, but implications for secondary CS teachers. The results of the study indicate that the use of pair programming in the Advanced Placement Computer Science Principles course can positively predict students' AP test score (Gray et al. 2019). The results also indicated that the students in the study with higher AP scores were taught by teachers with less than three

years of experience teaching CS. Therefore, in secondary CS education, pair programming not only helps students learn more, but helps teachers with less experience to have a greater effect in the classroom (Gray et al. 2019).

3.4 Other investigations of pair programming

3.4.1 Pairings During Pair Programming

In addition to investigating the effects of pair programming on K-12 students, research has been done to investigate ideal pairings when using pair programming collaboration methods. A study investigating pairings during pair programming determined that the success of a partnership compatibility depends on their prior programming experience (Zhong et al. 2016). However, when examining the correlation between previous results and capability, Zhong et al. did not find significant results in pairings with only girls (Zhong et al. 2016). In another study investigating pairings, Campe et al. found that the student's attitude toward collaboration and previous experience significantly influence collaborative behaviors (Campe et al. 2020). These studies were carried out specifically with elementary and middle school students. Therefore, the results may differ for high school students and may be influenced by other factors, such as friendship.

3.4.2 Affects of Pair Programming on Girls

A widely accepted result of pair programming studies is the benefit it serves for women and girls when learning programming. There are several studies that investigate this occurrence with girls in secondary education (Liebenberg et al. 2012; Tsan et al. 2016). A study investigating the effect of pair programming on girls' attitudes towards computing showed that they enjoyed the socialization of programming, which in turn led to greater persistence when solving problems (Liebenberg et al. 2012). Liebenberg et al. suggested to begin pair programming with girls as early as 10th grade to increase their interest in CS and also increase the chance that they would persist in CS (Liebenberg et al. 2012). In Tsan et al., the authors investigated the correlation between gender pairings in groups and student achievement in elementary CS education. The findings indicated that all pairs of females scored significantly lower on their artifacts compared to pairs with other gender groups (Tsan et al. 2016). This study specifically called for further investigation of pair programming, specifically through an equity lens.

3.4.3 Variations of Pair Programming

Analyses of the computing education literature have identified a lack of studies investigating variations of traditional pair programming roles (Salleh et al. 2010; Hanks et al. 2011; Umapathy and Ritzhaupt 2017). In a recent study, Tsan et al. investigated the effects of using two computers instead of one with elementary students, similar to the Puzzle method (Tsan et al. 2020). They found that students were resistant to input from their partner when only using one computer, but also felt detached from their partner when using two computers. Lytle et al. was the first to investigate the Puzzle method (described in Chapter 2), along with two other variations, "Separate" and "Together" (Lytle et al. 2020). Lytle et al. found students enjoyed the Puzzle method the most because it gave explicit guidelines on how students should collaborate with each other to complete several programming activities (Lytle et al. 2020). Although the researchers found Puzzle to be successful, they insisted that more work needed to be done to investigate the equity of the method (Lytle et al. 2020).

3.4.4 Pair Programming and Equity

To date, several studies have suggested that pair programming can create inequitable learning environments for K-12 students. (Shah et al. 2014) investigated a student with two different partners to determine how inequity could be established by using pair programming (Shah et al. 2014). In both pairings, Shah et al. found instances of inequity where one student positioned themselves to be more intelligent than the other. This greatly affected the ability of the other student to learn in each pair (Shah et al. 2014). Furthermore, (Lewis and Shah 2015) found that the speed with which students strive to complete an assignment is a major factor in determining if a student pairing is equitable (Lewis and Shah 2015). For example, if a student wishes to complete a task more quickly than their partner, this student would take over the activity and not listen to input from the other student. Given these short examples, more work is still needed to deeply investigate the nuances of equity in pairs, especially with the new collaboration technology available to students (Lewis and Shah 2015; Shah et al. 2014).

3.5 Discussion

Based on the previously discussed literature, we can see that there are discrepancies among the benefits of pair programming. Mostly what these benefits are and who benefits the most from this practice. The main concern from the literature is that there is an overall lack of understanding how pair programming affects students in K-12 settings, yet educators still practice this method of collaboration in the classroom. Although, there are benefits to this

practice in industry and post secondary settings, many of these benefits do not carry over to K-12 settings. One of the most obvious reasons being the previous experience and programming knowledge of K-12 students. Another may be the differing interpersonal relationships K-12 students have with each other compared to individuals who are coworkers. Finally, college age students and programming professionals are expected to have a certain level of collaboration skills; however, K-12 students have yet to develop these skills before they begin using pair programming. The question is raised then, is the traditional method of pair programming is the best method to use with these students, or is there a variation that could address these issues. In this work, I attempt to investigate whether a pair programming method, called Puzzle, that uses a Driver-Driver approach to collaboration would be better suited for K-12 students when learning to program.

3.6 Method Used In Dissertation

In Chapters 4 and 6 of this work, I will be using conversational analysis as a framework to analyze student behaviors. Conversational analysis is a way of understanding social interactions between individuals to reach a mutual goal or understanding. Conversational analysis is commonly used in the field of sociology to understand communication patterns, and has been previously used by CS Education researchers to assess instructors ideas about collaboration among elementary students (Zakaria et al. 2022), and student positioning during pair programming (Shah et al. 2014).

In Zarkaria et al., authors recorded pairs of students completing programming problems in the 2 C setting (Zakaria et al. 2022). This variation of pair programming used NetsBlox's concurrent editing function and allowed each student to have their own computer. During the study, students continued to take the traditional Driver-Navigator roles and switched at various time intervals (Zakaria et al. 2022). During these problems, the authors recorded the partners conversation and interactions. Using the coding scheme from Ruvalcaba et al. (Ruvalcaba et al. 2016), the authors coded the transcripts from the study. This coding scheme was developed for traditional pair programming methods with one computer. This caused there to be challenges when analyzing the students in the 2 C setting. To rectify these challenges, the authors modified the scheme to fit with their study (Zakaria et al. 2022).

Additionally, in Rodríguez et al., researchers investigated the conversations and learning gains of undergraduate students who used pair programming to complete programming activities (Rodríguez et al. 2017). The authors recorded students' conversations used an *a priori* codebook to analyze student's dialogue. The codebook included codes like "statements", "off-topic", and "yes/no question". The codebook used in this study is shared in Chapter 4.

Finally, in Shah et al., the authors investigated the conversations of primary students when using pair programming to collaborate while completing programming activities (Shah et al. 2014). Researchers used a conversational framework that analyzed student conversations at three different levels: distribution of talk, content of talk, and student positioning (Shah et al. 2014). The distribution of the talk refers to the amount of time each student participated in the conversation when collaborating. Next, the content of the talk refers to how the students talked to each other and the types of sentiment were said to each other. Finally, student positing refers to a student using tone of voice and short language to show they are the more intelligent partner in the pair.

In this work, I use the same conversational framework mentioned above to analyze student behavior when using the Puzzle method (Shah et al. 2014). However, I do not use the same coding scheme used in Zakaria et al. (2022). Instead, I use the coding scheme used in Rodríguez et al. because it was used to analyze talk of undergraduate computer science students (2017). Since my investigations focus solely on high school students, I deemed the coding scheme in Rodríguez et al. more appropriate in correctly analyzing high school students more developed level of language compared to elementary students (2017). Additionally, I chose to use the coding scheme of Rodríguez et al. because the students in that study completed larger programming activities than those in (Zakaria et al. 2022), which is more similar to studies in this work (2017). Similar to the methods of Zakaria et al. (2022), I modified the coding scheme of Rodríguez et al. to be more appropriate for the Puzzle method since the authors had investigated traditional pair programming. Details about these additions are discussed in Chapter 4 (2017).

CHAPTER

4

A CONVERSATIONAL ANALYSIS OF SECONDARY STUDENTS USING THE PUZZLE PAIR PROGRAMMING METHOD.

4.1 Introduction

Pair programming is a common collaboration method used in primary and secondary (K-12) computer science (CS) education (Zakaria 2021; Lytle et al. 2020). Traditional pair programming involves two students using one computer to complete a programming activity. Typically, one student in the pair physically programs - known as the "Driver" - while the other student gives the driver directions - known as the "Navigator" (Williams et al. 2000). In this method, students switch roles at specific time intervals chosen by the instructor with the goal of having each student in both roles for an equal amount of time. Traditional pair programming has been found to have many benefits for girls (Werner et al. 2004; McDowell et al. 2006), including increased confidence (Williams et al. 2000), increased enjoyment (Howard et al. 2006), and higher CS course completion rates (Braught et al. 2011).

However, previous work investigating learning gains of students when pair programming found that when students do not switch roles at all, the Navigator has significantly less learning

gains than the Driver (Rodríguez et al. 2017). Studies investigating pair programming with K-12 students have found that pair programming can lead to inequitable learning environments (Shah et al. 2014; Lewis and Shah 2015). In these situations, one student in the pair attempts to show they are more dominant and has a greater intellectual ability than the other student (Shah et al. 2014). This phenomenon is also known as student positioning since one student attempts to position themselves in the dynamic as more intelligent. For example, Lewis and Shah found when one student in a pair wishes to finish an assignment more quickly than another, this student positions themselves as more intelligent and attempts to gain more control than the other student (Lewis and Shah 2015). The inequity potential of traditional pair programming is significant because it is a common collaboration method used in K-12 CS education. If students are collaborating and having negative experiences, it may negatively impact student perceptions and enjoyment of CS, especially for young women. Which, in turn, can influence women's decisions to persist in CS (DuBow et al. 2017).

On the contrary, Tsan et al. attempt to address the dominance issues when pair programming by investigating the effects of pair programming with two computers instead of one (Tsan et al. 2020). The results on students' enjoyment and collaborative behaviors indicated students enjoyed using two computers more than one, however, some pairs felt disconnected from their partner (Tsan et al. 2020). Additionally, Lytle et al. investigated scaffolding to promote collaboration with secondary CS students using three variations of the traditional pair programming method in block-based programming (BBP) environments, such as Scratch or Snap! (Harvey et al. 2013) (Lytle et al. 2020). The results of this study indicate students preferred the "Puzzle" variation over the others (Lytle et al. 2020). During Puzzle, each student has a computer and the pair uses a BBP environment, NetsBlox (Broll et al. 2017), that allows users to concurrently edit the same file. Within the environment, the programming blocks are partitioned between the students in such a way that when all of the blocks are combined, it results in all of the available blocks in the environment. The partitioning of the blocks attempts to provide scaffolding for the students that promotes collaborative behaviors by encouraging discourse to complete an activity. Lytle et al. stated that more work is needed to investigate the full effects of the Puzzle method on students behaviors.

The purpose of this study is to expand upon the work of Lytle et al. by investigating the gender equity of the Puzzle pair programming method (2020). To assess the gender equity of the Puzzle method, we use conversational analysis to identify behaviors or spoken interactions that show one student attempting to dominate the other (Shah et al. 2014). Conversational analysis is a way of understanding social interactions between individuals to reach a mutual goal or understanding. Conversational analysis is commonly used in the field of sociology to understand communication patterns, and has been previously used by CS education re-

searchers to investigate collaboration among elementary students (Zakaria et al. 2022) and student positioning during pair programming (Shah et al. 2014).

4.2 Research Questions

In Chapter 4, I answer RQ1, How do students collaborate when using the Puzzle method?, which has the following subquestions:

- RQ1a What does the overall **distribution of student talk** look like when using the Puzzle method?
- RQ1b What are the most **frequent dialogue interactions** between students when using the Puzzle method? How does the frequency of these interactions differ between pairs?
- RQ1c How does the Puzzle method **support or deter students' attempts to position themselves** as having greater intellectual ability than their peer?

4.3 Background

There are several pair programming investigations involving primary (Iskrenovic-Momcilovic 2019; Bodaker and Rosenberg-Kima 2022; Wei et al. 2021; Tsan et al. 2020; Zakaria et al. 2022) and secondary (Werner and Denning 2009; Werner et al. 2009; Denner et al. 2021; Papadakis 2018; Gray et al. 2019) students. In the following section, I will describe relevant research in secondary educational settings that provides the basis for the current work in this paper.

4.3.1 Variations of Pair Programming

Analyses of the computing education literature have identified a lack of studies investigating variations of traditional pair programming roles (Salleh et al. 2010; Hanks et al. 2011; Umapathy and Ritzhaupt 2017). In a recent study, Tsan et al. investigated the effects of using two computers instead of one with elementary students, similar to the Puzzle method (Tsan et al. 2020). They found that students were resistant to input from their partner when only using one computer, but also felt detached from their partner when using two computers. Lytle et al. was the first to investigate Puzzle mode (described in Section 2.6), along with two other variations, "Separate" and "Together" (Lytle et al. 2020). Lytle et al. found students enjoyed the Puzzle method the most because it gave explicit guidelines on how students should collaborate with each other to complete several programming activities (Lytle et al. 2020). Although the researchers found

Puzzle to be successful, they insisted that more work needed to be done to investigate the equity of the method (Lytle et al. 2020).

4.3.2 Pairings During Pair Programming

In addition to investigating the effects of pair programming on K-12 students, research has been done to investigate ideal pairings when using pair programming collaboration methods. A study investigating pairings during pair programming determined that the success of a partnership compatibility depends on their prior programming experience (Zhong et al. 2016). However, when examining the correlation between previous results and capability, Zhong et al. did not find significant results in pairings with only girls (Zhong et al. 2016). In another study investigating pairings, Campe et al. found that the student's attitude toward collaboration and previous experience significantly influence collaborative behaviors (Campe et al. 2020). These studies were carried out specifically with elementary and middle school students and the results may differ for high school students.

4.3.3 Effects of Pair Programming on Girls

A widely accepted result of pair programming studies is the benefit it serves for women and girls when learning to program. There are several studies that investigate this occurrence with girls in secondary education (Liebenberg et al. 2012; Tsan et al. 2016). A study investigating the effect of pair programming on girls' attitudes towards computing showed that they enjoyed the socialization of programming, which led to greater persistence when solving problems (Liebenberg et al. 2012). Liebenberg et al. suggested beginning pair programming with girls as early as 10th grade to increase their interest in CS and increase the chance that they would persist in the field (Liebenberg et al. 2012).

4.3.4 Pair Programming and Equity

To date, several studies have suggested that pair programming can create inequitable learning environments for K-12 students. Shah et al. investigated a student with two different partners to determine how inequity could be established by using pair programming (Shah et al. 2014). In both pairings, Shah et al. found instances of inequity where one student positioned themselves to be more intelligent than the other. This greatly affected the ability of the other student to learn in each pair (Shah et al. 2014). Furthermore, Lewis and Shah found that the speed with which students strive to complete an assignment is a major factor in determining if a student pairing is equitable (Lewis and Shah 2015). For example, if a student wishes to complete a

task more quickly than their partner, this student would take over the activity and not listen to input from the other student. Given these short examples, more work is still needed to deeply investigate the nuances of equity in pairs, especially with new collaboration technology available to students (Lewis and Shah 2015; Shah et al. 2014).

4.4 Research Methods

4.4.1 Study Context

During this study, participants were paired together and given forty minutes to complete a BBP activity. All pairs completed the same activity. This study was facilitated over the course of one day during a high school internship program. Students who volunteered to be part of the study were paired using time availability. The NetsBlox environment allows users to share their programs with each other to collaborate concurrently (Broll et al. 2017); similarly to a document in Google Drive. The concurrent editing of programs is essential to the Puzzle method. While completing the activity, participants' dialogue was audio/video recorded for analysis.

4.4.2 Participants

Table 4.1: Self-identified genders of participants within each pair.

Pair	Participant Gender
1	Boy
	Boy
2	Girl
	Girl
3	Girl
	Boy
4	Boy
	Boy

The participants in this study were recruited from a software engineering themed internship program hosted by a CS research lab at a public land grant university in the Southeastern United States (Isvik et al. 2021; Catete et al. 2022).

There were eight total participants in this study, resulting in four pairs (see Table 4.1). The races of the participants were: East Asian/Pacific Islander/Asian Other (3), Southeast Asian/Indian (3), and Black/African-American (2). Demographic information of the participants has been reported separately to ensure anonymity.

All participants had prior experience programming and collaborating in BBP environments. However, previous collaboration involved the traditional Driver-Navigator pair programming method. This study took place during the last week of a 10 week internship program, therefore, it is possible students in certain pairs may have established relationships with each other before the study took place. When possible, we attempted to pair students together who had not worked with each other before.

4.5 Data Analysis

We analyzed the recorded conversations and partner interactions using conversational analysis (Shah et al. 2014). Conversational analysis is a way of understanding social interactions between individuals to reach a mutual goal or understanding (Hutchby and Wooffitt 2008). Conversational analysis is commonly used in the field of sociology to understand communication patterns, and has been previously used by CS Education researchers to investigate collaboration among elementary students (Zakaria et al. 2022) and student positioning during pair programming (Shah et al. 2014). Using this framing, we present the analysis of our Puzzle collaboration study.

4.5.1 RQ1a: Distribution of Student Talk

We used dialogue analysis to determine the distribution of talk among pairs (Stolcke et al. 2000). Distribution of talk refers to the frequency in which each student in a pair contributed to the dialogue. This method has been used in other studies investigating the content of dialogues between pairs during pair programming and in other CS collaboration studies (Vail and Boyer 2014; Rodríguez et al. 2017). To determine the frequency, we used the timestamps associated with participant dialogue from the audio transcriptions. The amount of time each participant contributed to the collaborative nature of the activity refers only to the dialogue between the participants. For example, when several of the groups were completing the activity, they conferred with the facilitator about the activity. This interaction is removed from distribution of student talk analysis to give greater insight into the pair's collaborative behaviors with each other.

4.5.2 RQ1b: Content of Student Talk

We used *a priori* tagging to determine the content of student dialogue. *A priori* analysis involves two or more researchers using previously defined tags to determine the frequency of specific verbal or physical actions of participants from transcribed audio. The *a priori* tags used to answer RQ1b are shown in Table 4.2. This qualitative method is frequently used when analyzing dialogue or interviews of participants. First, two researchers calculated an inter-rater reliability score of .97 for the *a priori* tagging process using one audio transcript. Next, the remaining transcripts were each tagged by two researchers. Finally, the researchers met to resolve any discrepancies from the tags. When investigating RQ1b, we examine the frequency of tags to investigate differences and similarities among all pairs.

Table 4.2: *A priori* tags used to analyze RQ1b.

Tag short hand	Definition
s	statement of information or explanation
u	opinion or indication of uncertainty
d	explicit instruction
su	polite or indirect instruction
ack	acknowledgment
m	meta-comment or reflection
qyn	yes/no question
qwh	wh-question (who, what , where, when, why, and how)
ayn	answer to yes/no question
awh	answer to wh- question
fp	positive task feedback
fnon	non-positive task feedback
o	off-task
ts	to self

4.5.3 RQ1c: Student Positioning

We used the *a priori* tags that were developed when investigating RQ1b to analyze student positioning, similar to the analysis done in (Shah et al. 2014). Here, we will focus on specific instances of conversation where one student positions themselves as being more intelligent than the other. For example, in situations where one student only gives directions and the other student only acknowledges the statement occurs frequently in a pair, this may be an indicator that they had an inequitable learning environment (Shah et al. 2014).

4.6 Results

In the following section we answer our subquestions, **RQ1a**) What does the overall distribution of student talk look like when using the Puzzle method?, **RQ1b**) What are the most frequent dialogue interactions between students when using the Puzzle method? How does the frequency of these interactions differ between pairs?, and **RQ1c**) How does the Puzzle method support or deter students' attempts to position themselves as having greater intellectual ability than their peer?, for each pair in this study. Following Barron's suggestion to focus on the group as the unit of analysis, the results for this study are presented in a case study fashion (Barron 2003).

4.6.1 Pair 1: M & M

Distribution of talk

This pair of students consisted of two young men: Student A and Student B. As shown in the left of Figure 4.1, Student B was more vocal (58.4%) in the interactions of the pair compared to Student A (41.6%). However, this pair had the most even distribution of talk compared to the other groups.

Content of talk

In the right of Figure 4.1, we can see the most common content tag for Student A is acknowledgment (ack). While the most common tag for Student B is to self (ts). To self (ts) refers to instances when students were talking to themselves through the directions of the activity or their code. It does not include dialogue that involves both students. An interesting result is the large amounts of indication of acknowledgment (ack) tags of Student A and the large amounts of statement (s) tags of Student B. When these two results are viewed together, it can be concluded that Student B may have taken a more dominant position in the pair by using

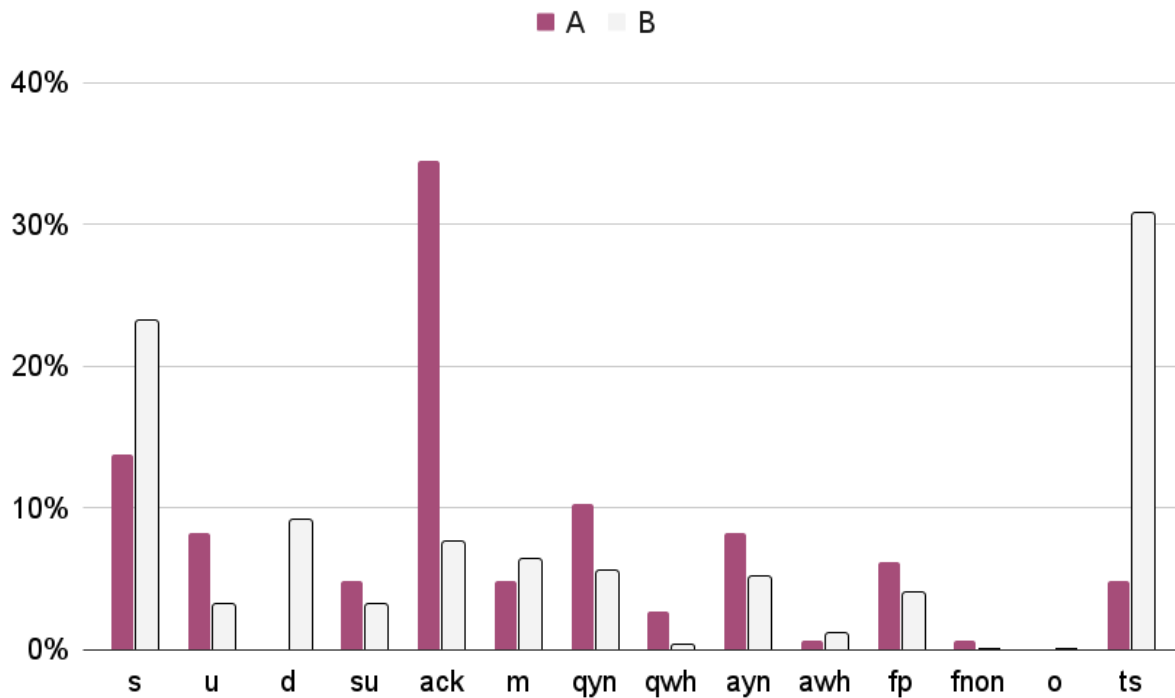


Figure 4.1: Percentage of each type of tag expressed during the activity for pair 1.

statements to communicate instead of questions. Student B also seemed to problem solve on his own which can be seen with the large about of to self (ts) tags.

Student positioning

During the activity, it was clear from the exchanges of the participants that Student B read the activity instructions and told Student A directions.

- B: Oh, you see where it is the call. Can you do the Google Maps and the get lat[itude] to thing again?
 A: Okay. Oh it's that for the first one, right?
 B: And get that X for the next one.
 A: Okay

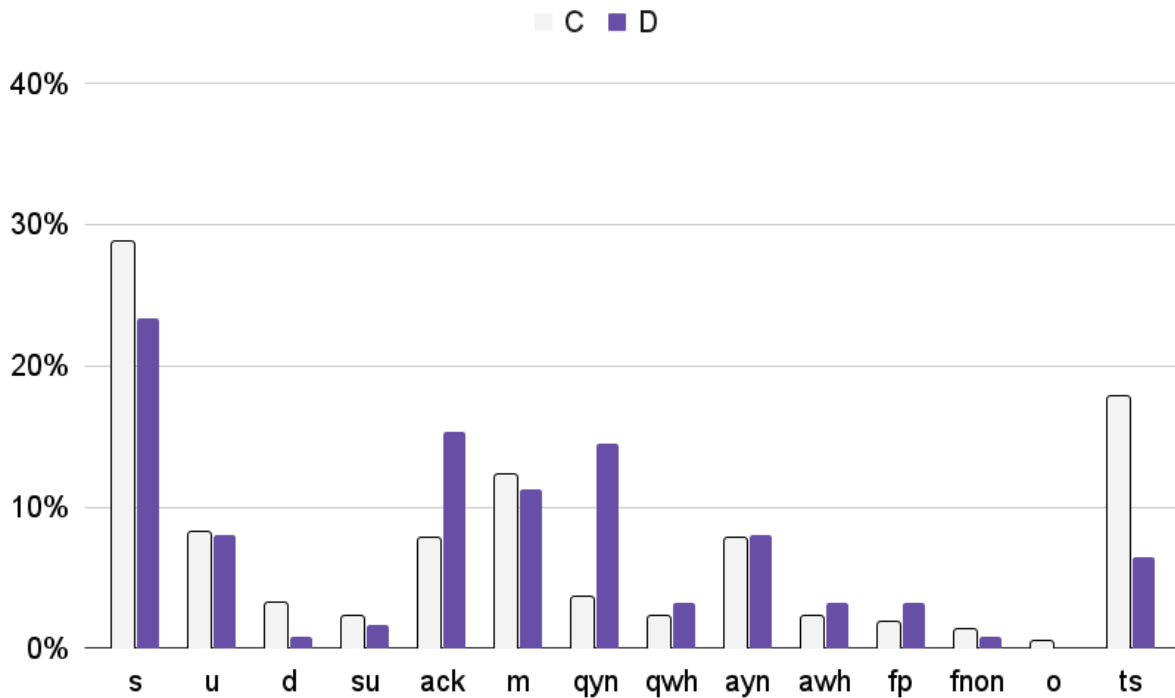


Figure 4.2: Percentage of each type of tag expressed during the activity for pair 2.

4.6.2 Pair 2: F & F

Distribution of talk

This pair of students consisted of two young women: Student C and Student D. As shown in the left of Figure 4.2, Student C was more vocal (69.2%) in the interactions of the pair compared to Student D (30.8%).

Content of talk

In the right of Figure 4.2, the count of tags said by each student is shown. Conversely to Pair 1, the most common tag for Student C and D was statements (s). However, Student C, the participant who contributed the most dialogue, also had a large amount of to self (ts) and meta-comment or reflection (m) tags compared to Student D. Student D communicated more frequently through questions and acknowledgment than Student C. This indicates Student C may have communicated more frequently with directions, while Student D mainly asked questions. The large amount of to self (ts) tags also indicates that Student C problem solved on her own, similar to Student B in Pair 1.

Student positioning

An example of Student C using statements to communicate is shown below:

- C: Can you see? Oh, okay. Oh, you're supposed to make that purple like when you made the [block]. Maybe I can edit it.
D: How do I set to make it in the custom thing, though?
C: Yeah, it says, making the custom thing and then choose purple and purple color.
D: I don't know if that really matters...

4.6.3 Pair 3: F & M

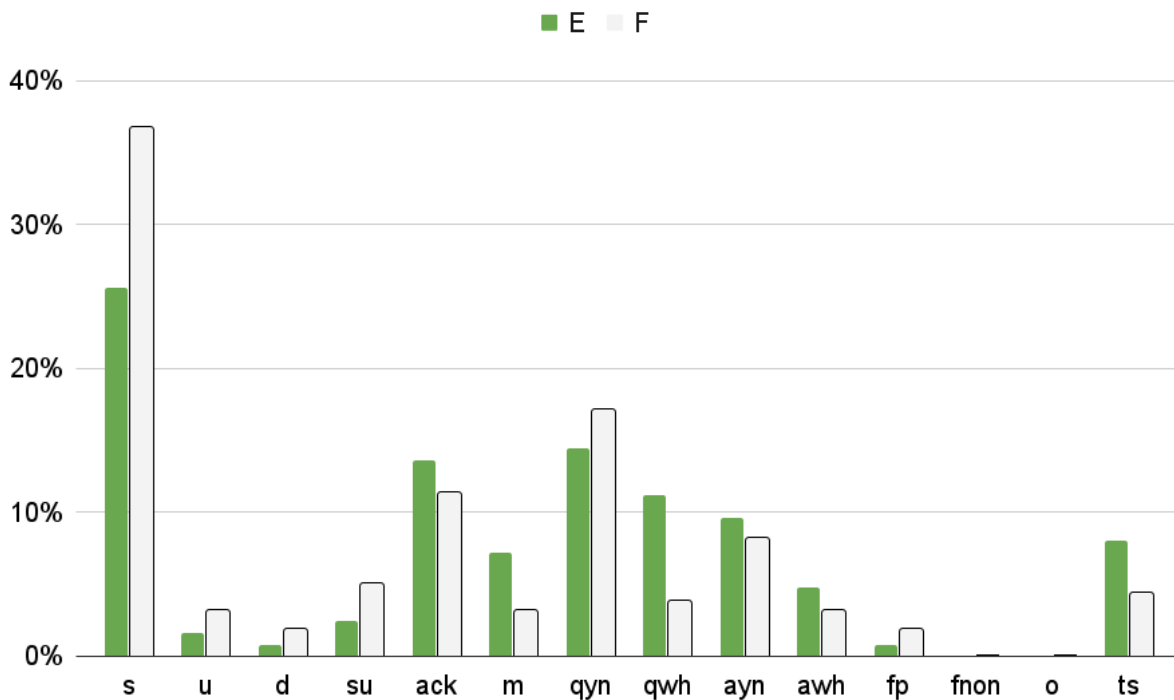


Figure 4.3: Percentage of each type of tag expressed during the activity for pair 3.

Distribution of talk

This pair of students consisted of one young man, Student E, and one young woman, Student F. Student F (69.7%) was more vocal in the interactions of the pair compared to Student E (30.3%)

Content of talk

In Figure 4.3, the count of tags said by each student is shown. What stands out in this figure is that both Student E and F have the least amount of to self (ts) tags than the other pairs in the study. Additionally, in contrast to the other pairs, both students in pair 3 used both yes/no questions (qyn) and wh-question (qwh) to collaborate. However, Student F had a large amount of statements (s) compared to Student E. These results indicate slightly collaborative behaviors from both students through the asking and answering of questions and choosing to work through problems together instead of individually.

Student positioning

When analyzing the transcript of pair 3, no student positioning occurred. However, there was an interesting exchange at the beginning of their interaction. Both Student E and Student F defined how they would collaborate at the beginning of the activity:

- E: Do you want like, maybe one person says the instructions and the other person does it?
- F: Yeah, we can do that. Like saying the instructions doing that?
- E: Okay, that'll be helpful. So open [the file] and name it Weather App
- F: So do you want me to say the instructions, and you do it? Or should I? You can say the instructions.
- E: Okay, yeah.

4.6.4 Pair 4: M & M

Distribution of talk

This pair consisted of two young men: Student G and Student H. As shown to the left of Figure 4.4, Student G (63.2%) was more vocal in the interactions of the pair compared to Student H (36.8%).

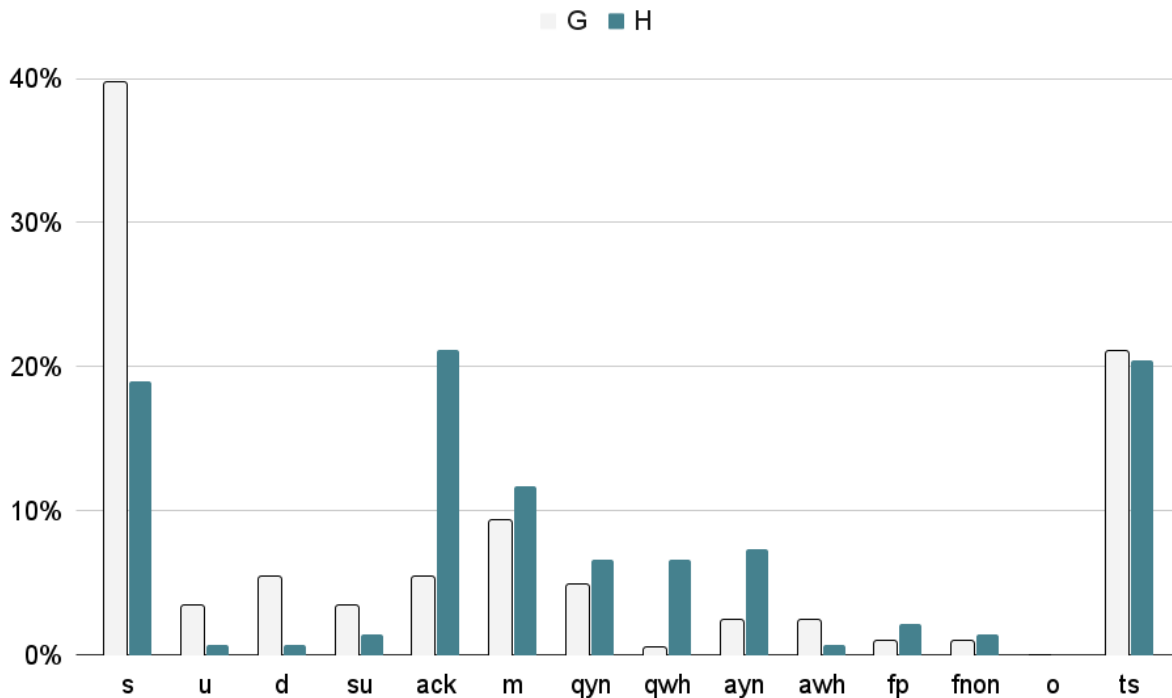


Figure 4.4: Percentage of each type of tag expressed during the activity for pair 4.

Content of talk

In the right of Figure 4.4, the count of tags said by each student is shown. Compared to the other groups in the study, Pair 4 had the least collaborative dialogue. Similarly to Pair 1, the student with the most dialogue, Student G, had the highest number of statement (s) tags. However, Student H had the highest amount of acknowledgment (ack) tags. This suggests that the pair communicated mostly through statements apposed to questions. An interesting result that is unique to Pair 4 is the amount of to self (ts) tags for Student H. In all of the previous groups, the student with the least spoken dialogue had little indication of to self (ts) tags; however, to self (ts) tags had the second highest count for Student H. These results show that both Student G and Student H preferred to solve problems individually, and when they had to collaborate, Student G was dominant.

Student positioning

We also observed strong student positioning with Pair 4. In the example below, we show an instance where Student G is showing intellectual dominance over Student H.

- G: [It] Doesn't let me duplicate from one place to another. What? You basically you need to put in the block that says -
- H: Sprite lat[itude] and sprite long[itude]
- G: No, no, it's that's right, like it's the. It's in the section variables. Then it's the block that it's like, set. It like sets a variable to a value.
- H: Oh, yeah, I I got what you say. I'm sorry. All right.

4.7 Discussion

In this study we investigated the collaborative behaviors of secondary students while using the Puzzle pair programming method. This study is the first of its kind to examine student behavior using a "Driver-Driver" pair programming method that defines student roles by partitioning block pallets between students in block-based programming environments. We were able to gain insight into student collaborative behaviors using a hybrid conversational analysis framework to analyze student conversations and verbal statements (Rodríguez et al. 2017; Shah et al. 2014). We analyzed the distribution of student talk (RQ1a), the content of student talk (RQ1b), and student positioning using *a priori* coding (RQ1c) of four pairs of students.

Our findings from RQ1a indicate that the overall distribution of talk when using the Puzzle method was not equal between any of the pairs in our study. This was surprising since we found pair 3 to behave more collaboratively. Furthermore, this led us to question what might be another method of determining the distribution of talk between students. It was determined that assigning different weights to different parts of the participants' dialogue in the calculation of time might be more insightful. This is because certain sections of conversations may hold greater significance in terms of the actions carried out by the participants.

Next, our findings from RQ1b suggest that there tends to be a pattern of dialog interactions between pairs that are less collaborative versus more collaborative. Pairs that were less collaborative had a dynamic where the more dominant participant said more statements, while the other participant said acknowledgments or asked questions. In these instances, the dominant participant also said a great amount of *to self* statements. The pair who had more collaborative behaviors, pair 3, communicated mostly through yes/no and wh-questions and statements. In addition, in pair 3, we saw the least number of *to self* tags from both participants. Therefore, we suggest that students use practices similar to interviewee think-aloud protocols that are designed to help others understand the thought processes of each person in the pairing (Hubbard et al. 2016). This will ensure that students communicate more openly about their plans and encourage collaborative behavior.

Finally, our findings from RQ1c show that the Puzzle method may not deter student attempts to position themselves as having a greater intellectual ability than their peer. Of the four pairs

in this study, there was evidence of student positioning in three. This could be due to previous relationships established through the internship program from which the participants were recruited. The most interesting finding was that the pair that did not have student positioning began their collaboration by further establishing their roles when starting the activity. We believe that this was the key to this pair's ability to collaborate successfully. In contrast, pairs who did not define their communication style or needs had less collaborative communication and mostly talked to themselves through problems instead of engaging their partner. CS pedagogy encourages students to plan their project development (McKinney and Denton 2006), however, this evidence supports the need to also plan collaboration practices, which may be especially useful for geographically separated or asynchronous teams.

4.8 Conclusion

The purpose of this study was to investigate collaborative behaviors of secondary students while using the Puzzle pair programming method. This study is the first of its kind to examine student behavior in a block-based programming environment with structured collaboration roles and concurrent editing. The results of this study show that the Puzzle method alone may not provide students with enough structure to collaborate effectively. However, the results do suggest with proper training and clearly defined roles, the Puzzle method may help students develop collaboration skills.

An implication of this study is how the Puzzle mode affected students' "Navigator" behavior while being in "Driver" roles. In the transcript excerpt from pair 1, we can see Student B explicitly giving instructions on what blocks Student A needs. In the traditional pair programming roles, it is common for students that are acting as the Navigator to try to take over the Driver's role of physically programming (Williams et al. 2000; Rodríguez et al. 2017). With the Puzzle method, we can see how students are forced to communicate since there is no physical possibility for students to access all the blocks they need to complete an activity. Thus, Puzzle may be a good training activity for learning how to be a good navigator for later collaborative activities.

A further implication of our findings is to provide training to students when introducing a new collaboration method. Before the study, all participants were shown an example of how to complete an activity using the Puzzle method. However, they were not given best collaboration practices. We hypothesize that students would have more collaborative behaviors if shown the best way to collaborate using the Puzzle method. Another implication for research is that when analyzing student dialogue, it is important to examine various factors of their dialogue holistically before coming to a conclusion. For example, if we had only taken the distribution of talk into account in determining the collaboration between the participants, we would never

have known that pair 3 was the more collaborative compared to pair 1.

Considerably more work will need to be done to determine the lasting effects of the Puzzle method on students' experiences and perceptions of CS. A natural progression of this work is to compare the behaviors of students when using traditional pair programming and the Puzzle method. Future studies should also investigate how student behavior changes over time while using the Puzzle method.

CHAPTER

5

STUDENT ATTITUDES AND PERCEPTIONS OF THE PUZZLE METHOD VERSUS TRADITIONAL PAIR PROGRAMMING

5.1 Introduction

In Chapter 4, we investigated student behavior when using the Puzzle method to complete a programming activity. In addition to student behavior, we must investigate the effect of the Puzzle method on student attitudes and perceptions. Measuring the effect on student attitudes and perceptions is important because they can affect student interest and intentions to persist in the field of computer science (CS) (Dou et al. 2020). Previous work investigating pair programming has found that it can positively affect student's self-efficacy (Wei et al. 2021), confidence (Braught et al. 2011), and attitudes toward computing (Papadakis 2018). There have also been investigations that show student attitudes can affect success of each pairing during pair programming (Zhong et al. 2016; Campe et al. 2020). However, there has yet to be a study examining the affect of the Puzzle method on student attitudes. In order to determine whether the Puzzle method is as beneficial to students as the traditional Driver-Navigator pair programming method, we must evaluate student attitudes while using Puzzle.

These effects will give us a holistic understanding of the Puzzle method and if students enjoy using it to collaborate. The purpose of this study is to investigate the effect of traditional and Puzzle pair programming on student attitudes, preferences of collaboration, and perceptions of their partner. We use a pre and post survey to measure the change in student attitudes towards collaboration, social support, computing confidence, and interest in programming. Students were paired together based on their initial attitudes towards collaboration and social support, and were separated into two groups, Puzzle and traditional. During this study students also completed a reflection after their activities with their partner to better understand their perceptions of collaboration.

5.2 Research Questions

In Chapter 5, I answer RQ2, How are student perceptions about collaboration and computer science affected by when using the Puzzle method compared to traditional pair programming?, which consists of the following subquestions:

- RQ2a) How are **students attitudes** affected by the Puzzle method and traditional pair programming?
- RQ2b) Do **students prefer** the Puzzle method compared to traditional pair programming?
- RQ2c) How do **student perceptions** of their partner affect their perceptions of collaboration?

5.3 Background

To date there have been several investigations of pair programming's affect on student attitudes at post secondary institutions. In Braught et al., authors investigated the effects of pair programming on students with low SAT scores (Braught et al. 2011). Their results indicate that students who pair program have higher confidence levels. In another study investigating pair programming, McDowell et al. found that it also increases student confidence, computer science proficiency, and improved program quality (Braught et al. 2011).

There has also been investigations into the methods affect on student attitudes in K-12 settings. In Wei et al., the authors investigated the self-efficacy and computational thinking skills of elementary students using "partial pair programming". This is described as letting students collaborate while completing a task as individual students (Wei et al. 2021). This was done to eliminate the monitoring the teacher would have to conduct to ensure that the

students switched roles at indicated intervals. Wei et al., found that the partial pair programming approach increases student self-efficacy and would be a beneficial collaboration method to use with students. Additionally, in a study investigating middle student attitudes towards collaboration, researchers found pairs where one student had a more positive attitude than their partner, their knowledge decreased (Denner et al. 2014). Finally, in Papadakis et al., the results indicated high school students' attitudes toward computing improved and that they also had improved comprehension of introductory programming concepts (Papadakis 2018).

5.4 Research Methods

In this section, I will describe the context of the study including activities students completed, all experimental measures, and the participants of the study.

5.4.1 Study Context

This study took place during the first three days of the Art, Coding, Action! summer camp at NC State (Nandwani 2022). The idea behind the camp is to teach novice programmers introductory programming concepts through block-based activities that use graphics and art. In total, there were 18 experimental hours during the study. On the first day of the study, participants will completed several activities that familiarized them with concepts used later in the camp. These activities included creating shapes, using various colors, and using loops to create patterns. On this day participants also completed a pre-survey after the camp introduction to gather information about their attitudes before pair programming (RQ2a).

On the second day of the study, participants were paired based on their self-assessed programming confidence and stated prior knowledge measured by the pre-survey. Half of the participants were assigned to the traditional pair programming group (control) while the other half were assigned to the Puzzle pair programming group (experimental). After the pairings were formed, students will completed two programming activities with their assigned partner and a reflection at the end of the day. This reflection questionnaire that asked about their and their partners collaborative behaviors for that day. Finally, on the last day of the study participants completed one programming activity with their partner and the post-survey individually.

During the camp, I divided the participants into two groups, traditional pair programming and puzzle pair programming. The traditional pair programming pairings will act as the control group as this is the common collaboration method used in K-12 CS classes (Zakaria 2021; Lytle et al. 2020). This will allow me to analyze the effect of the two different methods on student

attitudes. When participants did not assenting or receive parent consent, they were placed in the traditional pair programming group since the camp is originally designed to be taught using this method. Next I will describe all experimental measures associated with this study.

5.4.2 Participants

Table 5.1: Means of participant responses to attitudes toward collaboration and social support from the pre survey.

	Question	P	T
Attitudes Toward Collaboration (5)	I learn more when I work with others.	3.9	3.6
	I have more fun when I work with others.	4.1	4.3
	When more than one person works on a project, we have better ideas and make better things.	4.5	4.4
Social Support (4)	Important people in my life think its a good thing for me to learn about technology.	3.3	3.4
	I believe people like me can do well learning technology.	3.3	3.5
	My family like me to learn about technology.	3	3.4
	Other students generally think its cool that I learn about technology.	2.9	2.5

All participants were recruited from The Engineering Place summer programs at NC State, which charge a fee of \$500 and do not include transportation. Additionally all participants were entering the 9th or 10th grade. In the recruitment materials, there was no specification of programming experience needed to participate in the camp.

Participants completed the pre-survey on the first day of camp to ensure pairings had similar attitudes towards collaboration and social support about technology. On the second day of camp, the participants were paired based on their responses to the pre-survey. Meaning those with similar self-assessed attitudes were paired together. Using a Mann-Whitney U test (Fay and Proschan 2010), the non-parametric equivalent of the two independent sample t-test, I determined there were no statistically significant difference between the control group's and the experimental group's attitudes towards collaboration and social support (see Table 5.1). In total this resulted in 9 pairs, 5 being in the experimental group and 4 being in the control group. Of the participants, 15 self-identified as boys, and 3 self-identified as girls.

5.5 Data Analysis

In the section below, I describe the constructs measured in the pre and post survey, the questions included in the reflection questions, and the process in which I used thematic analysis to analyze the open-ended responses from both instruments.

5.5.1 Pre and Post Survey

The pre survey was distributed on the first day of the camp before any activities were started. While the post survey was given on the the last day of camp before the "show and tell" activity. Both survey's were distributed to participants through the online survey software Qualtrics. There will be a total of 4 sections on the pre survey and 5 on the post survey. The constructs measured from the pre survey are:

- Attitude toward collaboration ($\alpha = .83$) (Barron et al. 2014),
- Social support ($\alpha = .76$) (Weston et al. 2019),
- Computer confidence ($\alpha = .71$)(Barron et al. 2014),
- Programming interest ($\alpha = .85$) (Weston et al. 2019),

The first two constructs, Attitude toward collaboration, and Computer confidence, have a 5 point scale Likert value questions, with 5 being the highest value. While the last two constructs, Programming interest and Social support, have a 4 point scale (Not at all, Only a little, Pretty much, and A lot) for Likert value questions, with 4 being the highest value. All questions from the pre and post surveys are located in Table A.1 located in Appendix A.

The post-survey contains all constructs from the pre-survey in addition to a section about perceptions and collaboration with their partner. Sections in the post survey asking about participants perceptions and collaboration with their partner were specifically added to the post survey to give us insight into how the Puzzle and traditional pair programming methods affected these items. The questions in this section were derived from the Friendship Quality scale and were modified to be appropriate for this study (Bukowski et al. 1994) (see Table A.1 in Appendix A). All Likert value questions are on a 5 point-scale with 5 being the highest value ($\alpha = .972$).

5.5.2 End Reflection

Participants completed daily reflection "exit tickets" at the end of each day to assess their self-perceived abilities through out the camp. These exit tickets were distributed through Qualtrics

during the last 15 minutes of camp every day. The questions on the exit tickets are as follows:

R1 Do you think you were a collaborative partner today? Why or why not?

R2 Do you think your partner was collaborative today? Why or why not?

R3 What are actions you could take to be more collaborative in the future?

The questions on the exit tickets were adapted from the National Center for Women and Information Technology Pair Programming Student Final Assessment (for Women & Information Technology 2016). The first question, Q1 assesses how each participant felt about their collaborative behaviors. The second question, Q2, assesses how each participant felt about their partners' collaborative behaviors. Finally, Q3 assesses which collaborative behaviors participants want to enact in the future.

5.5.3 Thematic Analysis

To gain insight into the students open question responses from the reflections and the post survey, myself and another researcher used thematic analysis. Thematic analysis involves a team of researchers coding qualitative data, then condensing that data into common themes. This methodology is often used to analyze open response questions from participant surveys (Maguire and Delahunt 2017). A qualitative approach gives us additional insight into participants thoughts and attitudes towards each pair programming method that we may not get from the quantitative data. First, two researchers, including myself, established a norm for tagging student responses by tagging 5 responses together and then discussing our similarities and differences. Then, after agreeing on the first set of codes, we coded the remaining student response. Next, we combined alike codes and grouped the codes into themes. Finally, we ensured that all codes were only mapped to one theme and that all themes had definitions.

5.6 Results

In this section, I address the results for each of the subquestions in Chapter 5: RQ2a) How are **students attitudes** affected by the Puzzle method and traditional pair programming?, RQ2b) Do **students prefer** the Puzzle method compared to traditional pair programming?, and RQ2c) How do **student perceptions** of their partner affect their perceptions of collaboration?

5.6.1 RQ2a: Student attitudes

The aim of RQ2a is to determine the change of participant’s attitude toward collaboration, and programming interest. During the study discussed in this chapter, I distributed a pre and post survey to students on the first and last day of the study, respectively.

First, I determined that the data from the survey was not normally distributed using the Shapiro-Wilk test (Shapiro and Wilk 1965), which is a calculation that determines if a data set has normal distribution. Next, I conducted a Wilcoxon signed rank test to determine if there were statistically significant changes in student responses from the pre survey to post survey (Hollander et al. 2013). This statistical test is the non parametric test equivalent of the paired t-test. I used this statistical test since the data was not normally distributed and interval. I did not observe statistically significant results from the Wilcoxon signed rank test.

I also performed the Mann-Whitney U test on student responses to the post survey. The Mann-Whitney U test is the non-parametric equivalent of the 2 independent sample t-test (Fay and Proschan 2010). I used this statistical test since the data was not normally distributed and interval. In these calculations, students in the Puzzle (experimental) group were Sample 1, while students in the traditional (control) group were Sample 2. The results from this test also indicated no statistically significant difference between the two groups. In Tables 5.2, 5.3, 5.5, and 5.4, I display the mean of the two groups of students for each question in the pre and post survey.

Table 5.2: Mean values of student responses from the pre and post survey from both groups about attitudes toward collaboration.

Attitudes Toward Collaboration (5)	Puzzle		Traditional	
Question	Pre	Post	Pre	Post
Q1. I learn more when I work with others.	3.9	4.0	3.6	4.0
Q2. I have more fun when I work with others.	4.1	4.6	4.3	4.1
Q3. When more than one person works on a project, we have better ideas and make better things.	4.5	4.4	4.4	4.0

Although there were no significant results found in the survey, the means of the participant responses can give us insight into how the method they used affected their attitudes towards collaboration (see Table 5.2. For Q1, the responses for the traditional and Puzzle groups increased. In the Puzzle group, the responses for Q2 increased as well, but Q3 decreased. However, in the traditional group, responses for Q2 and Q3 decreased. Overall, this indicates students may

Table 5.3: Mean values of student responses from the pre and post survey from both groups about participants' social support with technology.

Social Support with Technology (4)	Puzzle		Traditional	
Question	Pre	Post	Pre	Post
Q4. Important people in my life think it's a good thing for me to learn about technology.	3.3	3.2	3.4	3.3
Q5. I believe people like me can do well learning technology.	3.3	3.4	3.5	3.3
Q6. My family likes me to learn about technology.	3.0	3.3	3.4	3.3
Q7. Other students generally think it's cool that I learn about technology.	2.9	3.0	2.5	2.6

feel they learn more when working with others but may not feel that they make better things. These results also show that students in the Puzzle group may have enjoyed their collaborative experience over students in the traditional group.

Table 5.4: Mean values of student responses from the pre and post survey from both groups about participants' confidence with computers.

Confidence with Computers (5)	Puzzle		Traditional	
Question	Pre	Post	Pre	Post
Q8. I am good with computers.	3.8	4.2	3.9	3.8
Q9. I feel confident about my ability to use computers.	4.0	3.9	4.1	4.0
Q10. When I use technology I think about how it works.	3.9	3.9	3.5	3.6
Q11. I like the challenge of learning how to use a new technology.	4.3	4.1	3.6	3.9
Q12. I feel like I belong in computer science.	3.5	3.4	3.0	3.4
Q13. I consider myself to be a computer programmer.	2.6	3.0	2.8	3.4
Q14. I take pride in my computing abilities.	2.8	3.2	3.4	3.8

When looking at the student responses in Table 5.4, we can see slight increases in participant computer confidence in the Puzzle group (Q8, Q13, Q14) and the traditional group (Q12, Q13, Q14). For Q8, participant responses from the Puzzle group increased, however responses from the traditional group decreased. While the opposite occurred in each group for Q12. However, for both Q13 and Q14, responses in both groups increased. For Q13, responses increased by .4 (Puzzle) and .6 (traditional) and for Q14, responses increased by .4 in both groups. The results for Q13 and Q14 are interesting because they suggest an increase in student self-efficacy in

Table 5.5: Mean values of student responses from the pre and post survey from both groups about participants interest in programming.

Interest in Programming (4)	Puzzle		Traditional	
Question	Pre	Post	Pre	Post
Q15. How interested are you in programming computers or other technologies (in other words writing code)?	2.9	3.1	3.1	2.9
Q16. How much do you want to learn how to program computers or other technologies?	3.2	3.1	2.9	3.1
Q17. How much do you want to learn computer science?	3.1	3.1	3.1	2.8

computer science.

Although the results do not indicate statistically significant differences in participants' pre and post survey responses or the responses between the two groups, we gain insight into how the participants' attitudes changed, particularly their attitudes toward collaboration and computer confidence. Participant's attitudes toward collaboration became more positive after participating in the study. The participants in the Puzzle group felt they had more fun while participants in the traditional group felt they learned more. Participants in both groups also had more confidence in their abilities with computers, indicating an increase in self-efficacy as well.

5.6.2 RQ2b: Student Preferences

Table 5.6: Count and student responses from Q# on the post survey which asked participants "Did you find the pair programming (Puzzle or traditional) method to be helpful when completing the programming activities?"

Selection	Puzzle		Traditional	
	n	%	n	%
Yes	4.0	40%	0	0%
Somewhat	2.0	20%	4.0	50%
No	4.0	40%	4.0	50%
Total	10		8	

To gain insight into the student's preferences between traditional pair programming and Puzzle pair programming (RQ2b), the post survey asked "Did you find the pair programming

Table 5.7: Themes found from the open responses in the post survey relating to traditional and Puzzle pair programming.

Puzzle	
Theme	Definition
Not Helpful	Students did not find the Puzzle method helpful when collaborating.
Enjoyed Puzzle	Students found the Puzzle method enjoyable when collaborating.
Communication	Students found that the Puzzle method made them communicate with their partner compared to other experiences collaborating.
Traditional	
Theme	Definition
Preferred Traditional	Students preferred traditional pair programming when collaborating.
Confusing	Student found traditional pair programming confusing.
Technical Difficulties	Students experienced technical difficulties.

method to be helpful when completing the programming activities?". In Table 5.6, we display the count and percentages of participants for each possible answer: Yes, Somewhat, and No. Interestingly, 40% of the participants in the Puzzle group responded "Yes", while 0% of the participants in the traditional group said "Yes". However, 40% of the participants in both groups said "No". The participant's explanations for their responses give more insight into their responses.

Using thematic analysis, myself and another researcher analyzed the participants' responses which resulted in six themes, three for each group. For the Puzzle group, the themes found were: Not Helpful, Enjoyed Puzzle, and Communication. For the traditional group, the themes found were: Preferred Traditional, Confusing, and Technical Difficulties. In Table 5.7 we show the definitions of each theme for each group.

Several participants in the Puzzle group felt that Puzzle was not helpful while others expressed their enjoyment with the method. One participant that found Puzzle not helpful because it restricted their ideas and said, "[Puzzle] was not helpful to me because it restricted me from properly expressing my ideas in the form of code as I was not able to use some of the code block types and would not be able to recover these [i]deas due to the restrictions." Another student faced technical difficulties when attempting to collaborate, "When we tried to work at the same time, the program would stop syncing us and that was very inconvenient." One student who enjoyed Puzzle stated, "The Puzzle method was helpful when completing the programming activities because I feel like it helped us interact with each other and get our ideas across much easier. We finished a lot of work together because of it, and we enjoyed our

time working on projects together." Another student echoing the same sentiment said, "It was good because we could learn to 'solve' the puzzle."

The most intriguing theme found in the analysis of the Puzzle group was Communication. In the Puzzle group, one participant said, "I thought it was very cool to do the things like the puzzle mode, because it taught me how to collaborate better." Another participant said, "I think it was useful in terms of improving communication skills." However, one student discussed collaboration, but in a negative connotation, "It was sort of difficult to do something because we had to communicate ALOT." The amount of participants that discussed communication is interesting because the overall goal of the Puzzle method is to increase the communication between pairs.

Participants in the traditional group expressed that they enjoyed using the traditional method, one participant said, "I think it was helpful because we were able to work on the same work and put both of our ideas onto one screen and we were able to fix each others mistakes." However, another participant expressed that it was confusing, "I had a little bit of trouble understanding everything." Finally, the participants faced several technical difficulties with NetsBlox itself which could contribute to the more negative attitudes towards traditional pair programming, "we had technical difficulties and that made it way harder to work together we went on our own most of the time."

Overall, the student responses from the post survey explain as to why more participants in the Puzzle group found it to be more helpful than participants in the traditional group. The most intriguing insight is the sentiment that Puzzle encouraged participants to collaborate, however, that may not be the most enjoyable collaborative method for everyone. Another interesting result is that when participants used the Puzzle method they felt restricted because their options in the bbp environment were not as easily at their disposal compared to their previous programming experiences.

5.6.3 RQ2c: Student Perceptions

To gain insight into how student pairings may have affected participants' attitudes and preferences toward each collaboration method (RQ2c), in the post survey we asked them about how they felt about their partner. In Table 5.8, we display the mean likert-value responses of each group from the post survey (scale of 5). From these results, we can see that the participants did not have strong negative attitudes toward their partners. However, there were slightly more positive responses from the students in the Puzzle group ($M = 4.1$) than the traditional group ($M = 3.9$). In addition to asking about the participant's feelings towards their partner in the post survey, we also asked about how they felt the collaboration with their partner went

Table 5.8: Mean values of student responses from the post survey from both groups about participants views of their partners.

Question (5)	Puzzle	Traditional
I feel happy when I am with my partner.	4.2	4.3
If my partner stopped coming to camp I would miss them.	4.1	3.9
When I do a good job at something my partner is happy for me.	4.1	4.1
Sometimes my partner does things for me or makes me feel special.	4.2	3.9
I can count on my partner for help.	4.5	4.4
I think about my partner even when my partner is not around.	4.0	3.4
I consider my partner to be a friend.	4.3	4.0
My partner and I spend time together outside of this class.	3.6	3.4

in a reflection questionnaire after completing the activities with their partners (see Section 5.5.2). All of these questions were open responses and were analyzed by two researchers using thematic analysis.

In total, there were 11 themes, 5 for the Puzzle group, and 6 for the traditional group. The definitions of each theme are displayed in Table 5.9. When asked, R1) "Do you think you were a collaborative partner today? Why or why not?", two themes emerged from the responses of participants in the Puzzle group: Completed Activities and Technical Difficulties. In relation to completed activities, one participant said, "I think that I was a very collaborative partner today because I was able to work effectively with my peer and was able to complete the required task in a effective and efficient way." Another participant said: "I think so because we were... able to complete most of the challenges." When discussing technical challenges, one participant said, "[N]o because the whole system was frustrating because of the glitches, so i just did it by myself." While another participant echoing the same sentiment said, "No, we had technical difficulties, so we had trouble." From the participant responses in the traditional group, there were also two themes: Contributed to Work and No Collaboration. When discussing their contribution to their work, one participant said: "Yes I was a collaborative partner because I contributed heavily to the game-making effort by doing the majority of the coding." However when discussing the lack of collaboration, one participant stated, "Not really, there was not much collaboration", and another said, "I think we both shared concepts and ideas to each other, but I'd think we weren't working together. I usually just get honed in to the coding."

Next, when asked, R2) "Do you think your partner was collaborative today? Why or why not?", one theme developed from the responses of participants in the Puzzle group: Communicated with Partner. Regarding communication with their partner, one participant said, "I think so because we both communicated with each other and we had a lot of good results.", and another

said, "I believe that my partner was collaborative today because he was able to effectively solve any problems or issues we had through effective communication and good use of problem-solving process." From the participant responses in the traditional group, there were two themes: Worked Well Together and Effective Communication. When discussing working with their partner, a participant stated, " My partner was collaborative because he had good and effective communication that greatly contributed to the development of the projects.", and another said, "Very. They did their best to help communicate any ideas or concepts I was stuck on." When discussing working well with their partner, one participant said, "Yes, we were both doing the same thing so we were on the task together so we could work together."

Finally, when asked, R3) "What are actions you could take to be more collaborative in the future?", two themes emerged from the responses of participants in the Puzzle group: Improve Communication and Get to Know Each Other. When discussing the need to improving communication with their partner one participant said, "I could be a more collaborative partner in the future by initiating more communication having more discussions with my partner." When discussing getting to know their partner more, one participant said, "I think that we could. be more collaborative by also getting to know each other a little bit before we started so that we wont be awkward for the beginning of our project." and another participant said, "I think we can be more collaborative by getting to know each other first and do introduction." From the participant responses in the traditional group, there were three themes that emerged: Improve Communication, Check Partner Understanding of Instructions, and Stay on Task. When discussing the need to improve communication, one participant said, "One of the actions that I could take to be more collaborative in the future is by communicating more with my partner.". Next, one participant said, "I probably could of asked my partner is they were confused by any of the coding that I was or wasn't doing.", when discussing checking in with their partner. Finally, when discussing staying on task, one participant said, "I could focus better and we could stay on task more."

In summary, when participants were asked about their perception of being collaborative partners, two prominent themes emerged from the Puzzle group responses: Completed Activities and Technical Difficulties. Participants in this group highlighted their collaborative nature when tasks were completed efficiently but also expressed frustration and decreased collaboration due to technical challenges. Conversely, participants in the traditional group explained their contributions to work and lack of collaboration as prominent themes.

Furthermore, when participants were asked about their perception of their partners' collaboration, one theme, Communicated with Partner, arose from the Puzzle group responses, emphasizing effective communication as a marker of collaboration. In contrast, the traditional group responses highlighted themes of Working Well Together and Effective Communication,

suggesting that participants in this group valued mutual understanding and communication skills.

Regarding actions for future collaboration, the Puzzle group emphasized the need to Improve Communication and Get to Know Each Other, highlighting the importance of establishing familiarity with their partners and clear communication channels. Similarly, the traditional group identified themes of Improve Communication, Check Partner Understanding of Instructions, and Stay on Task, indicating the importance of effective communication, clarity in task comprehension, and maintaining focus.

5.7 Discussion

Using a pre and post survey as well as a reflection, we collected data on the participants' attitudes, preferences, and perceptions of collaboration from two groups, Puzzle and traditional. The results of the pre and post survey indicate participant attitudes toward collaboration slightly improved and their computing confidence increased in both groups. We also found that more participants preferred Puzzle collaboration compared to the traditional method. Finally, we found that how student's perceived their partner did not have an effect on their collaboration method preference.

The findings of RQ2a indicate that there were positive attitude shifts for students in both groups relating to attitudes toward collaboration and computing confidence. The two questions that showed increased confidence were, (Q13) " I consider myself to be a computer programmer." and (Q14) "I take pride in my computing abilities." The change in confidence for these specific questions are interesting because they are also related to student identity in computer science. This can foster a sense of belonging (Yang et al. 2021) and community (Dahn and DeLiema 2020), which in turn could lead to interest or retention of students in CS (Burg et al. 2015; Capobianco 2006). These findings suggest that the implementation of the Puzzle method in CS education may positively impact students' attitudes towards collaboration and their confidence in their computing abilities.

Next, the findings of RQ2b indicate that the students preferred the Puzzle method more than the students who preferred the traditional method. When we looked closer at why this was occurring, it seemed that some students liked the level of collaboration that Puzzle encouraged. However, there were also negative views on the subject. We also saw that the students found puzzle restrictive, while traditional allowed them to be more creative. The most intriguing finding from this analysis is that the purpose of Puzzle was fulfilled in forcing students to communicate more when collaborating; however, not all students enjoyed how much communication was required to be successful. Therefore, in line with the findings in Chapter 4, we

suggest that Puzzle be used as a training tool to teach best practices to students when beginning to learn about collaboration. Then, after some familiarity with how to best collaborate with a partner, allow students to choose their preferred method of collaboration.

Finally, the findings of RQ2c indicate that there was little relationship between the participants' perceptions of their partner and their perceptions of each collaboration method. However, we found that there were similarities and differences in perceptions of collaboration between the two groups. When discussing their own collaboration, participants in both groups used their accomplishments and completed activities as metrics of collaboration. Similarly, when discussing their partners' behaviors, participants in the Puzzle group believed that their partners were working collaboratively based on their outcomes, not on their behaviors. However, participants in the traditional group considered communication and understanding as attributes of successful collaboration of their partners. These findings suggest that the Puzzle method may make students place a higher value on producing and accomplishing their end goal while the traditional method encourages students to place value on communication. Although the overall goal of the Puzzle method is to encourage students to communicate while collaborating.

There are several limitations to this work, including the unequal number of participants in each group and the short duration of the study. There were unequal numbers of participants in each group because a handful of students at the camp chose to participate in the study. The short duration of the camp was due to the availability of participants. However, the data and results we were able to conclude from this study still provide the computing education community with insights into the effects of the Puzzle method.

5.8 Conclusion

The purpose of this study is to investigate the effect of traditional and Puzzle pair programming on student attitudes, preferences of collaboration, and perceptions of their partner. We found evidence that the Puzzle method can positively affect student attitudes toward collaboration and increase students' computing confidence. We also concluded that the Puzzle method maybe used best as a training tool to teach collaboration practices to students while also learning how to program. We suggest, after some familiarity with using the Puzzle method and displaying behaviors of successful collaboration, to allow students to choose their preferred method of collaboration. We think it is important for educators to consider the preferences and needs of their students when choosing between collaborative methods such as Puzzle and traditional methods, as some students may prefer a more role structured approach, such as driver-navigator, while others may thrive in an environment that emphasizes communication,

such as Puzzle.

A natural progression of this work is to implement a replication study with a greater number of participants in a formal classroom environment. The formal classroom environment will provide us with more context of how the relationships between partners can affect collaboration with the Puzzle method. It will also help us understand how using the Puzzle method in an established learning environment can affect students' attitudes towards computer science and programming. Future studies should also consider investigating student learning gains and self-efficacy when students use the traditional method compared to when they use the Puzzle method.

Table 5.9: Themes found from the open responses in the reflection relating to participant's and their partners collaboration behaviors.

Puzzle	
Theme	Definition
Complete Activities	Students in the Puzzle group expressed they were a collaborative partner because their paring was able to complete many of the activities.
Technical Difficulties	Students in the Puzzle group expressed they thought there collaboration was not successful because of technical difficulties.
Communicated with Partner	Students in the Puzzle group explained their partner was collaborative because they communicated with their partners to complete the activities.
Improve Communication	Students in the Puzzle group stated they could be more collaborative in the future by communicating more.
Get to Know Each Other	Students in the Puzzle group stated they could have tried to "get to know" their partner better.
Traditional	
Theme	Definition
Contributed to Work	Students in the traditional group expressed they were collaborative because they contributed to the work being done in their pairs.
No Collaboration	Students in the traditional group expressed there was no collaboration between them and their partner.
Effective Collaboration	Students in the traditional group explained their partner was collaborative because they were able to effectively communicate.
Improve Communication	Students in the traditional group stated they could be more collaborative in the future by communicating more.
Check Partner Understanding of Instructions	Students in the traditional group stated they could be more collaborative in the future by checking in with their partners understanding of the instructions.
Stay on Task	Students in the traditional group stated they could be more collaborative in the future by staying on talk and completing their activities.

CHAPTER

6

CONVERSATIONAL ANALYSIS AND ATTITUDES OF STUDENTS USING THE PUZZLE METHOD VERSUS TRADITIONAL PAIR PROGRAMMING

6.1 Introduction

In previous chapters, we investigated student behavior when using the Puzzle method and differences in student attitudes and perceptions of computer science when using Puzzle and traditional pair programming. Our previous work has identified training may be necessary for students to be successful when using the Puzzle method and that students perception of each collaboration method may be heavily dependent on their preferences of communication. However, we have yet to explore the differences in student behavior when using the traditional method compared to the Puzzle method. The purpose of this chapter is to investigate the differences in student behavior and determine student preferences of collaboration method after experiencing both traditional and Puzzle pair programming. We use the same conversational analysis framework described in Chapter 4 to analyze student conversations and conduct

semi-structured interviews with study participants to gain insight into their views of each method.

6.2 Research Questions

In Chapter 6, I answer RQ3, How do student behaviors differ when students use the traditional pair programming method compared to the Puzzle method?, which has the following subquestions:

- RQ3a) How does the **distribution of talk** differ when students use traditional and Puzzle pair programming?
- RQ3b) How does the **content of talk** differ when students use traditional and Puzzle pair programming?
- RQ3c) How does **student positioning** differ when students use traditional and Puzzle pair programming?
- RQ3d) After experiencing both traditional and Puzzle pair programming, **which method do students prefer?**

6.3 Background

In this research paper, conversational analysis serves as the primary framework for examining student behaviors during pair programming activities. Conversational analysis, a method commonly employed in sociology to dissect communication patterns, has also been utilized in the field of Computer Science (CS) Education research. Previous studies, such as Zakaria et al. (2022) and Rodriguez et al. (2017), have applied conversational analysis to assess collaboration dynamics among students during programming tasks.

Zakaria et al. (2022) investigated student behaviors in the 2 C setting, a variation of pair programming where each student has their own computer. This study utilized a modified coding scheme adapted from Ruvalcaba et al. (2016), originally designed for traditional pair programming with a single computer. Similarly, Rodriguez et al. (2017) explored student conversations and learning gains during pair programming activities among undergraduate students, utilizing an *a priori* codebook to analyze dialogue.

Furthermore, Shah et al. (2014) examined the conversational dynamics of primary students engaged in pair programming activities. Their study employed a conversational framework focusing on talk distribution, content, and student positioning. Building upon these previous

works, the work in this chapter adopts a similar conversational framework to analyze student behaviors when using the Puzzle method. However, instead of the coding scheme used in Zakaria et al. (2022), the coding scheme from Rodriguez et al. (2017) is used due to its alignment with the higher level of language expected from high school students and the complexity of programming activities involved. Modifications to the coding scheme are implemented to ensure its suitability for analyzing Puzzle method collaborations. Detailed discussions of these modifications are provided in Chapter 4.

6.4 Research Methods

The purpose of this study is to investigate the differences in student behavior when using traditional pair programming and the Puzzle method to complete programming activities. I also investigate student perceptions of each collaboration method and suggestions that can be made to the programming environment to improve collaboration for students.

There were two participants in this study, both self-identified as male. One participant, Partner 1, is in the 11th grade, and Partner 2, is in 12th grade. The participants had a unique relationship in that they are friends, but do not attend the same schools. Therefore, they have not worked together before in an academic setting. Before the study, both participants had experience with block-based programming but did not have experience with pair programming or collaboration when programming.

We recorded our two participants over Zoom completing two programming activities, one with the traditional method and one with puzzle. During the traditional pair programming activity, Partner 1 acted as the driver throughout the entirety of the activity while Partner 2 acted as the navigator. On the second day of the study, the pair completed another programming activity but using the Puzzle method. Throughout the activity Partner 2 had access to the following block pallets in the Puzzle activity: network, control, looks, motion, sound. While Partner 1 had access to the following block pallets: variables, operators, sensing, custom, pen. To expand on the findings of Chapter 4 and 5, we also provided both participants with a training video involving traditional pair programming. In this video, students were shown the role definitions of Driver and Navigator, as well as best practices for collaboratively programming. This includes respecting your partners ideas, communicating with your partner, and not taking over the activity.

In Figures 6.1 and 6.2 are screen shots of the BBP environment for Partner 1 and Partner 2 during the Puzzle activity. On the final day of the study, the participants participated in semi-structured interviews in which I asked them questions concerning their opinions of the collaboration methods and changes they would make to the block-based programming

environment to make collaboration easier. The questions asked in these interviews are located in Appendix C.

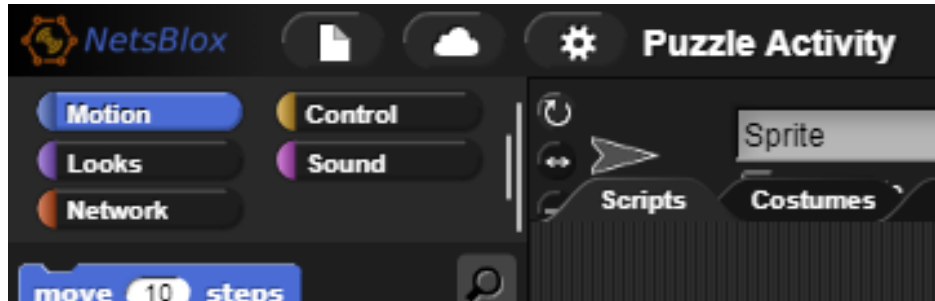


Figure 6.1: Screenshot of Partner 1's available block palettes.

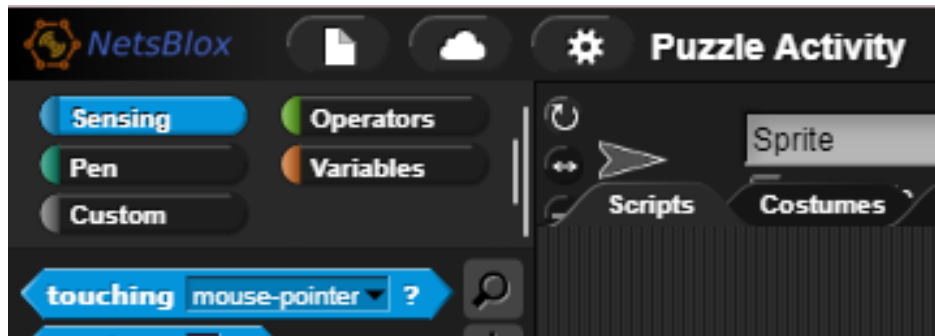


Figure 6.2: Screenshot of Partner 2's available block palettes.

6.5 Data Analysis

In this section we describe the analysis methods we used to analyze the data of this study. We analyzed the recorded conversations and partner interactions using the same conversational analysis used in Chapter 4. We analyzed participant interviews using the codes that were created in Chapter 5 during the thematic analysis. More details of our processes are described in below.

6.6 Conversational Analysis

We use the same analysis methods used in Chapter 4 to analyze participant conversations in the chapter. In summary, we examined three elements of their talk: distribution of talk, content of talk, and student positioning. Distribution of talk refers to the frequency in which each student in a pair contributed to the dialogue. This method has been used in other studies investigating the content of dialogues between pairs during pair programming and in other CS collaboration studies (Vail and Boyer 2014; Rodríguez et al. 2017). Next, content of talk refers to the type of language participants used to communicate with each other during the activities. We used *a priori* tagging to determine the content of the student's talk. *A priori* analysis involves two or more researchers using previously defined tags to determine the frequency of specific verbal or physical actions of participants from transcribed audio. The *a priori* tags used to answer RQ3b are shown in Table 6.1 (Rodríguez et al. 2017). Finally, student positioning is the instances of conversation where one student positions themselves as being more intelligent than the other. We used the *a priori* tags that were used when investigating RQ1b to determine student positioning, similar to the analysis performed in (Shah et al. 2014).

Used codes and themes from Chapter 5 to do the thematic analysis because it shows a continuation of patterns and can be compared to other responses.

6.6.1 Thematic Analysis

We used thematic analysis to determine themes that existed in Partner 1 and Partner 2's interviews about each collaboration method. To establish a continuation of patterns, myself and another researcher used the tags developed from the analysis completed in Chapter 5 to analyze participant interviews. Using the tags from Chapter 5, we saw that Partner 1 and Partner 2's responses are strongly associated with results found in the previous chapter.

Table 6.1: *A priori* tags used to analyze RQ3b.

Tag short hand	Definition
s	statement of information or explanation
u	opinion or indication of uncertainty
d	explicit instruction
su	polite or indirect instruction
ack	acknowledgment
m	meta-comment or reflection
qyn	yes/no question
qwh	wh-question (who, what , where, when, why, and how)
ayn	answer to yes/no question
awh	answer to wh- question
fp	positive task feedback
fnon	non-positive task feedback
o	off-task
ts	to self

6.7 Results

In this section, I address the results for each of the subquestions in Chapter 6: RQ3a) How does the **distribution of talk** differ when students use traditional and Puzzle pair programming?, RQ3b) How does the **content of talk** differ when students use traditional and Puzzle pair programming?, RQ3c) How does **student positioning** differ when students use traditional and Puzzle pair programming?, and RQ3d) After experiencing both traditional and Puzzle pair programming, **which method do students prefer?**

6.7.1 RQ3a: Distribution of Talk

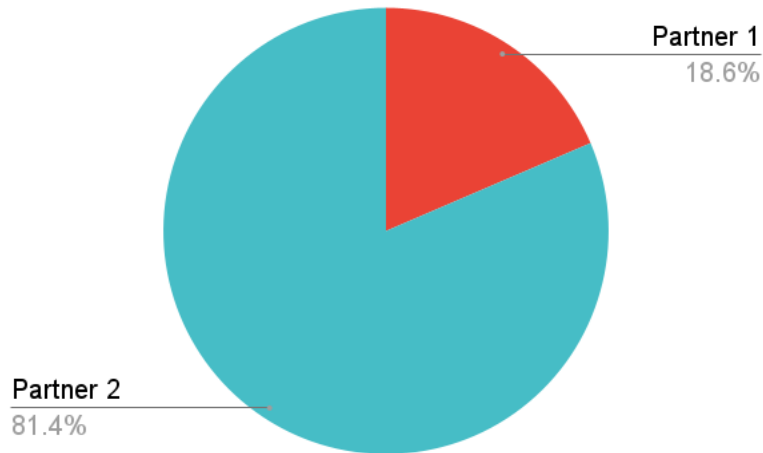


Figure 6.3: Distribution of talk between Partner 1 and Partner 2 while using traditional pair programming.

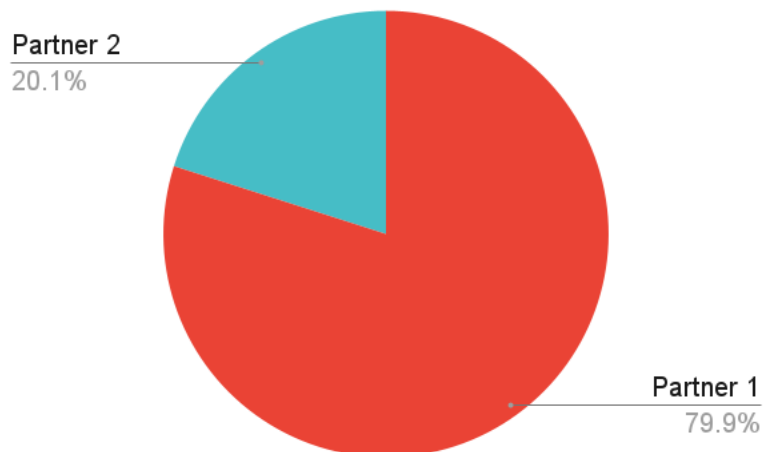


Figure 6.4: Distribution of talk between Partner 1 and Partner 2 while using Puzzle pair programming.

In Figure 6.3, we display the distribution of talk of Partner 1 and Partner 2 while using the traditional pair programming method. The results shown here are expected, since one partner typically contributes more communication when acting as the navigator, Partner 2. In contrast, we can see the distribution of talk of the pair while using the Puzzle method in Figure 6.4. We can see that Partner 1 is contributed the most amount of verbal communication when using the Puzzle method. The possible causes of why this occurred are discussed in Section 6.8.

6.7.2 RQ3b: Content of Talk

Traditional Pair Programming

In Figure 6.5, we display the percentage of each tag Partner 1 and Partner 2 expressed while completing a programming activity with the traditional pair programming method. The most obvious result is the use of "explicit instruction", or D, during the activity. This is expected because of the structure of traditional pair programming where the Navigator gives instructions to the Driver about what to physically program. We can also see a low number of "to-self" tags, or TS, which indicates more collaborative interactions between students because they are conversing with each other, not only thinking aloud.

Additionally, in Figure 6.6, we show the number of tags each participant expressed during the traditional pair programming activity. From the figure, we can see that Partner 2, who acted as the navigator, said the most "explicit instruction" directions. This is also consistent with the behavior expected from the Driver-Navigator dynamic. Another result consistent with the traditional pair programming method is that partner 1 asks more questions than partner 2, and partner 2 responds.

Puzzle Pair Programming

In Figure 6.7, we display the percentage of each tag Partner 1 and Partner 2 expressed while completing a programming activity with the Puzzle pair programming method. One of the first notable differences between Figure 6.5 and Figure 6.7 is the number of "explicit instruction" tags. This may have been caused by the less restrictive partner structure of the Puzzle method, since both Partners are expected to communicate their contribution to the activity, not only give instructions, such as the role of the navigator in the traditional method. During the Puzzle activity, each participant contributed more to the conversations while completing the activity, which also caused there to be an increase of the types of communication used to communicate. We can see this in Figure 6.7 with an increase in "wh-questions" (5%) and "meta-comment or reflection" (9%), and "acknowledgments" (13%).

Additionally, in Figure 6.6, we show the number of tags each participant expressed during the Puzzle pair programming activity. The first notable difference between Figure 6.6 and Figure 6.8 is the amount of comments, expressed in the number of tags, Partner 1 made compared to Partner 2. Another difference is during the Puzzle activity, neither partner expressed any "to-self" tags. However, in Chapter 4, we saw that at least one partner in each pair expressed "to-self" comments when completing their activities. The lack of these tags from Partner 1 and Partner 2 may be caused by the Puzzle activity being their second time collaborating together. By completing the traditional activity first, they were able to understand how each other best

collaborated and implemented those actions during the Puzzle activity.

6.7.3 RQ3c: Student Positioning

When examining student positioning between the two partners during the traditional pair programming method we saw this instance:

Partner 2: ... Then change zoom by one and then add in the update map block after.
Just right under.

Partner 1: Yes, but plus there, I think if there's 2 of them like, I think the total work there's already like it's like. I think ...

Partner 2: Oh, we just keep it, for now and then we'll see what happens.

Here, Partner 1 is trying to address a concern he had during the program because there were blocks with similar meaning and he did not know why there were two. Instead of Partner 2 explaining why there were two of the same block, he dismissed Partner 1's concern and said that would "see what happens" in the program. In this example, Partner 2 is attempting to position himself as more intelligent by disregarding the question of Partner 1. During the Puzzle method we did not see any instances of student positioning. However, we did see that Partner 1 was more dominant in the conversation than Partner 2, which is also supported from the distribution of talk shown above.

6.7.4 RQ3d: Student Preferences

We conducted semi-structured interviews to gain insight into each participant's opinion and enjoyment of each collaboration method. In summary, we found that Partner 1 preferred using the traditional pair programming method while Partner 2 preferred using the Puzzle method. As a reminder, Partner 1 was the driver during the traditional activity, while Partner 2 was the navigator.

Partner 1

When discussing why he preferred the traditional method, Partner 1 said, "I feel like, honestly, I felt more comfortable, and, like I learned, more using [traditional] than [Puzzle]", and expanded on this by saying, "I'd say I prefer the driver navigator more cause I was able to like cause I was a driver. I'd like, learn more. And it was just like more collaboration between me and [Partner 2]." When asked what he disliked about the traditional method, he said: "I'd say. Maybe like

I felt I learned a lot, but I don't know if [Partner 2] learned as much because he was only the person seeing the instructions, so I don't know if he got as much information or knowledge that I did." Finally, when asked about how he felt about the collaboration with Partner 2 during the traditional method, Partner 1 said, "I thought, like the communication was good cause. I was getting like clear instructions, and I wasn't like super confused on like what [Partner 2] actually meant. The communication was good enough to where I could be like. Oh, so if I need this line of command, I know exactly where to look for, and I'm not confused between what [Partner 2] meant and like what [Partner 2] was implying, or things like that. It was like direct communication, which was nice."

Next, when discussing collaborating with the Puzzle method, Partner 1 said, "I feel like the second method was a little more chaotic because we were trying to figure out what sections we had, and especially if they were code where it's involved. " However, Partner 1 also communicated he felt like himself and Partner 2 developed their communication skills while using Puzzle, "we definitely develop some communication skills more because it was like a lot more of like both of us were like evenly working together." Partner 1 provided an example of this and explained a scenario where the participants had to communicate more using the Puzzle method: "we were like able to give me like, communicate that like, oh, I need this block, and you have, like you have it in under one of your sections, so you should drop it down and like connect it here so that I can like finish the rest of the commands and like run the code." When asked what Partner 1 disliked about Puzzle he said, "I feel like [traditional] was just more clear, as like what I had to do and what he had to do, like the instructions and what our jobs, where are more clear for the [traditional] method than the puzzle [method]."

Additionally, I asked Partner 1 "Which method do you feel like you were able to focus more on the concepts being used in the activity not just the programming?" He said that he felt that way with the traditional method and that, "I was just able to understand the knowledge ... or the information more ... knowing what blocks [I was] specifically given". When asked which method he would use in the future, Partner 1 said, "I would choose a I driver navigator method, most likely [b]ecause I just was able to grasp more information, and it was a lot more like clear, direct communication with my partner, and we didn't have any confusions or things like that." In general, Partner 1 expressed a preference for the traditional method of collaboration, because of a greater level of comfort and perceived learning advantages over the Puzzle method. He highlighted the driver-navigator dynamic as conducive to his learning style, enabling clearer communication and a deeper understanding of the programming instructions.

However, he voiced concerns about the unequal distribution of learning opportunities between himself as the driver and Partner 2 as the navigator. Conversely, while Partner 1 acknowledged the development of communication skills during Puzzle collaboration, he found

it more chaotic and less clear compared to the traditional approach. Despite recognizing improved communication, he emphasized the clarity and directness of communication in the traditional method. Partner 1 expressed a preference to focus on conceptual understanding over programming specifics, feeling that the traditional method allowed for a clearer grasp of the concepts involved. Ultimately, he indicated a preference for the traditional method in future collaborations due to its perceived clarity, direct communication, and conducive learning environment.

Partner 2

When Partner 2 was asked about how he felt about the traditional method, he said, “I felt like that was fine, but I didn’t feel like it was that effective.” Expanding on this opinion he said, “Since I was a navigator. It was just kind of like explaining what to do. Not really like doing anything hands on, you know, like, with the code cause I was just telling [Partner 1] what to do, what not to do.” He also said, “[P]ersonally, I like doing things like hands on. So I didn’t really like, think it was that effective. Because when I feel like when coding and stuff, it’s better that you do like code as well. Cause it’s better for you, and you learn more and get more out of it.” When asked about the communication using driver-navigator, Partner 2 said, “I felt like it was pretty good, though the communication wise. I think he was able to understand everything that I was saying, and telling him to do the thing in that aspect. It was, good.” Next, when asked about the Puzzle method, Partner 2 said, “I felt like the puzzle method was better than I was. I got more out of it to cause we both like had like a hands-on experience, and we both had to like work through problems together, cause we wouldn’t have the same like, we don’t have the same blocks so we both had to work together and collaborate to get to like the end product.” I also asked if he enjoyed having separate blocks and he said, “I did think that was a good thing cause both of us had to work together, and so it was still like where we had to tell each other what to do, but not like. But we also had an opportunity to do the code ourselves.”

When asked about the communication when using the Puzzle method, Partner 2 said, “So for the puzzle method, I did think it was a little harder communicating, even though, like, you know, I felt like doing things hands-on was better. It’s like I felt like it was a little harder communicating, since we had different like blocks that we like assigned to us, so we both had to like do it like we had to wait. And like, you know, we if one person didn’t have it, we’d have to can communicate that to them and say, like, say, I don’t have this one, you have that. And then I felt like it did take a little longer than [traditional]. Finally, when asked which method he would want to use in the future, he said, “I think I would still choose the puzzle method, even though communication was a little harder. I felt like we got more out of it, and we were able to like do it together pretty much cause we both were working on the code together. We could

see we could both see what was happening. It was not just one person doing all the code, one person telling them what to do.” To clarify, when Partner 2 says that the communication was “harder”, he meant that the communication was more involved and with each other instead of one person mainly communicating while using the traditional method.

Partner 2 expressed mixed feelings about the traditional method of collaboration, finding it less effective due to his role as a navigator, which limited his hands-on involvement in coding. While he acknowledged effective communication, he believed that hands-on experience with coding was essential for better learning outcomes. In contrast, Partner 2 preferred the Puzzle method, appreciating the hands-on experience and collaborative problem-solving it facilitated. Despite finding communication slightly more challenging due to the need to coordinate different blocks, he valued the shared experience of working through challenges together. Partner 2 chose the Puzzle method for future pair programming, emphasizing the benefits of shared coding tasks and the opportunity for both partners to actively engage in the coding process. Despite acknowledging communication challenges, he perceived the Puzzle method as more conducive to collaborative learning and joint problem-solving.

Similarities

There are several similarities between the sentiments of Partner 1 and Partner 2. The first similarity is a preference for hands-on learning. Both partners express a preference for methods that allow them to actively engage in coding tasks rather than solely instructing or being instructed. Another is the value of communication. Partner 1 and Partner 2 recognize the importance of effective communication in collaboration, whether it is providing clear instructions or coordinating tasks with their partner. Next, their attitudes toward collaboration; they both appreciate collaborative problem solving and working together to achieve the end product, indicating a shared belief in the benefits of teamwork. Despite their differing preferences for collaboration methods, both partners acknowledge the effectiveness of certain aspects of each method, such as communication clarity in the traditional method and hands-on experience in the Puzzle method. Overall, while they may have different preferences for collaboration methods, Partner 1 and Partner 2 share common perspectives on the value of hands-on learning, effective communication, collaboration, and method effectiveness in pair programming.

Differences

The main difference between the opinions of Partner 1 and Partner 2 about collaboration methods is that Partner 1 preferred traditional, while Partner 2 preferred Puzzle. The main reason Partner 2 preferred Puzzle is because of the hands-on experience it afforded him com-

pared to when he was the navigator in traditional pair programming. Although typically in driver-navigator pair programming, the driver and the navigator switch as timed intervals or after activities, this is not always ensured to happen in the classroom. One of the main purposes of the Puzzle method was to provide a collaboration technique that would evenly divide tasks to students in order to have equal effort between students, but not require them to physically switch roles at specific intervals. Thus, increasing the instructor mental load and ability to help students in their classroom. The preferences of Partner 1 and Partner 2 after experiencing both methods provide evidence that Puzzle may be a more helpful collaboration tool in the classroom, since both students would be able to gain hands-on experience compared to the traditional method, which does not always grant each student an equal amount of time as the driver.

6.8 Discussion

In this study, we use the same conversational analysis framework used in Chapter 4 to analyze partners' conversations with each other. We also interviewed each student about their opinions on each type of method. We found that the students behaved more collaboratively using the Puzzle method by engaging in various types of dialogue. However, we also found that Partner 1 preferred the traditional method, and Partner 2 preferred the Puzzle method.

The findings of RQ3a indicate that the students behaved as expected when using the traditional method. However, they did not have an even distribution of talk during the Puzzle method. When looking through the lens of the student responses to the interview questions, we can see that Partner 2 preferred Puzzle, but thought the communication was "hard". This may indicate why he did not contribute as much as Partner 1. This also may indicate that Partner 2 placed a higher value on being able to have hands-on experiences compared to 'more effective' communication.

Next, the findings of RQ3b indicate that Partner 1 and Partner 2 behaved more collaboratively using the Puzzle method. There was an overall increase in the types of conversations they used to engage each other which included questions, meta comments, and acknowledgments when using the Puzzle method. This may have been caused by the lack of communication restrictions Puzzle puts on students. The traditional method specifically states that the navigator is to give instructions to the driver, which in turn, caused the pair to communicate using mostly explicit instructions. However, the Puzzle method may encourage students to ask more questions and communicate with each other more like a conversation.

The findings of RQ3c indicate that dominance in each of the collaboration methods remains even when the pairings are familiar with each other. During the traditional pair programming,

Partner 2 attempted to position himself as more intelligence by dismissing Partner 1's concerns. However, during the Puzzle activity, Partner 1 took on a more dominant role by being the one to read the instructions and tell Partner 2 what blocks were needed to complete the program.

Finally, the findings of RQ3d indicate that the use of different collaboration methods in programming activities can lead to differences in student preferences and perceptions. However, both students still value the importance of collaboration and teamwork in achieving the end goal. Partner 1 preferred using the traditional method because of the clear instruction provided by Partner 2, the navigator, and the hands-on experience as the driver. While Partner 2 preferred the Puzzle method also because of its hands-on learning for each participant and the engagement between himself and Partner 1. The preferences of Partner 1 and Partner 2 after experiencing both methods provide evidence that Puzzle may be a more helpful collaboration tool in the classroom, since both students would be able to gain hands-on experience compared to the traditional method, which does not always grant each student an equal amount of time as the driver. In Rodríguez et al., the authors show that the role of the navigator does not provide the same benefits as the driver role. These findings support this work in showing that the role of navigator may not be the best way for some students to learn how to collaborate when programming.

The main limitation of this study is the low number of participants. Although there were only two participants involved in the study, the in-depth analysis of their behaviors gives us insight into the differences in student behavior when using traditional and Puzzle pair programming. These results give us insight into, after experiencing both methods, potential student preferences and perceptions of collaboration.

6.9 Conclusions

The purpose of this study was to investigate the behavioral differences of students when using traditional pair programming and Puzzle pair programming. We also sought to understand student preferences for each method and their reasoning. We recorded two students completing two different programming activities, one using driver-navigator collaboration and the other using Puzzle. We found that the students behaved more collaboratively using the Puzzle method by engaging in various types of dialogue. However, each preferred a different collaboration method in the end.

This study provides the computing education research community with suggestions for future research and implications for using Puzzle in the classroom. When using Puzzle in the classroom it is important to consider certain aspects of pairing students, such as experience level, attitudes, and also how they prefer to communicate. In attempts to avoid one student

dominating the collaboration, we suggest pairing students together who have similar communication levels. One of the most important things to remember about introducing computer science (CS) to K-12 students is the feeling of belonging and interest in the field. By pairing students with similar communication preferences together, it is more likely the students will feel a sense of community, which can lead to interest and persistence in CS.

In future research, we suggest changing the order of the collaboration method. For example, students would use Puzzle first and then the traditional method. We also suggest investigating the engagement consistency of students during traditional and Puzzle pair programming. It is often noted that the navigator role can often 'check-out' in traditional pair programming, whereas the limitations of blocks in Puzzle would necessitate both partners being engaged throughout. Repeating the study with role changes occurring periodically or even by switching partners/blocks could investigate this theme.

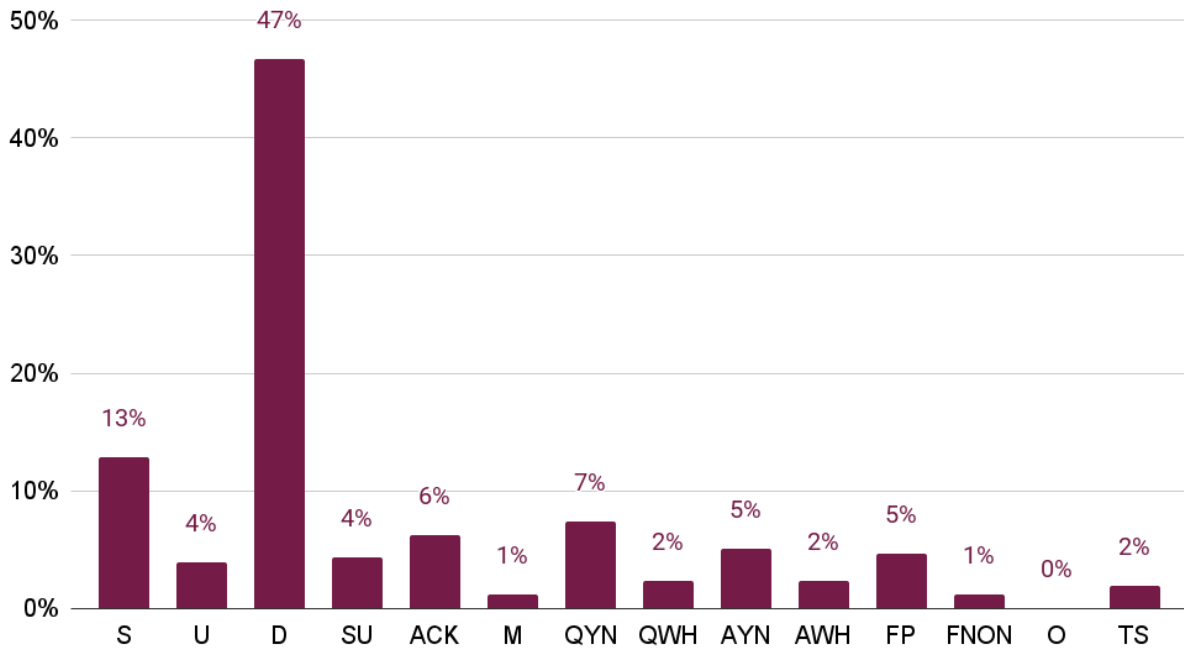


Figure 6.5: Percentage of tags during the activity completed with the traditional pair programming method

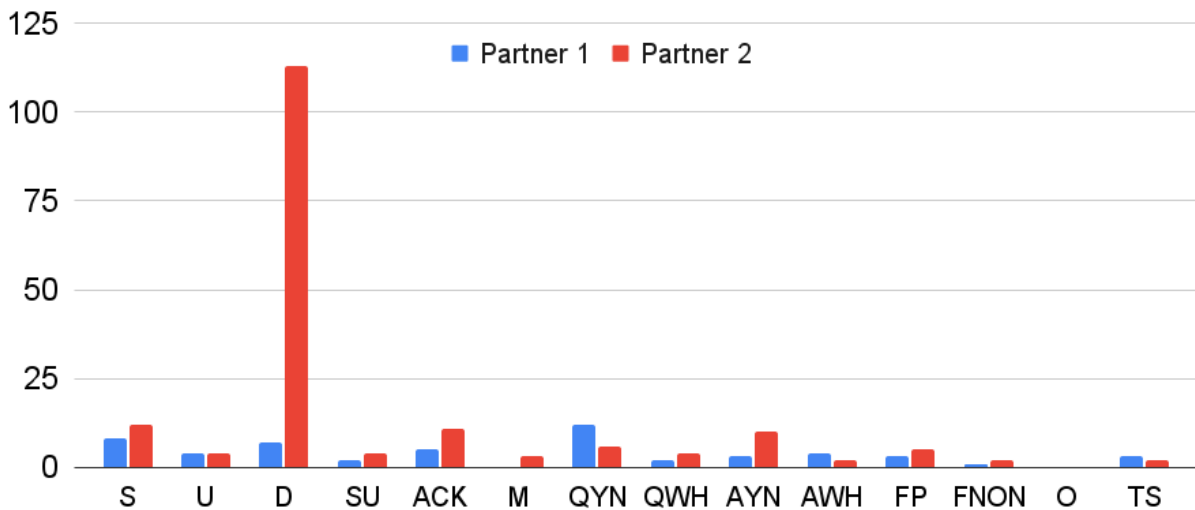


Figure 6.6: Count of tags for each participant during the activity completed with the traditional pair programming method

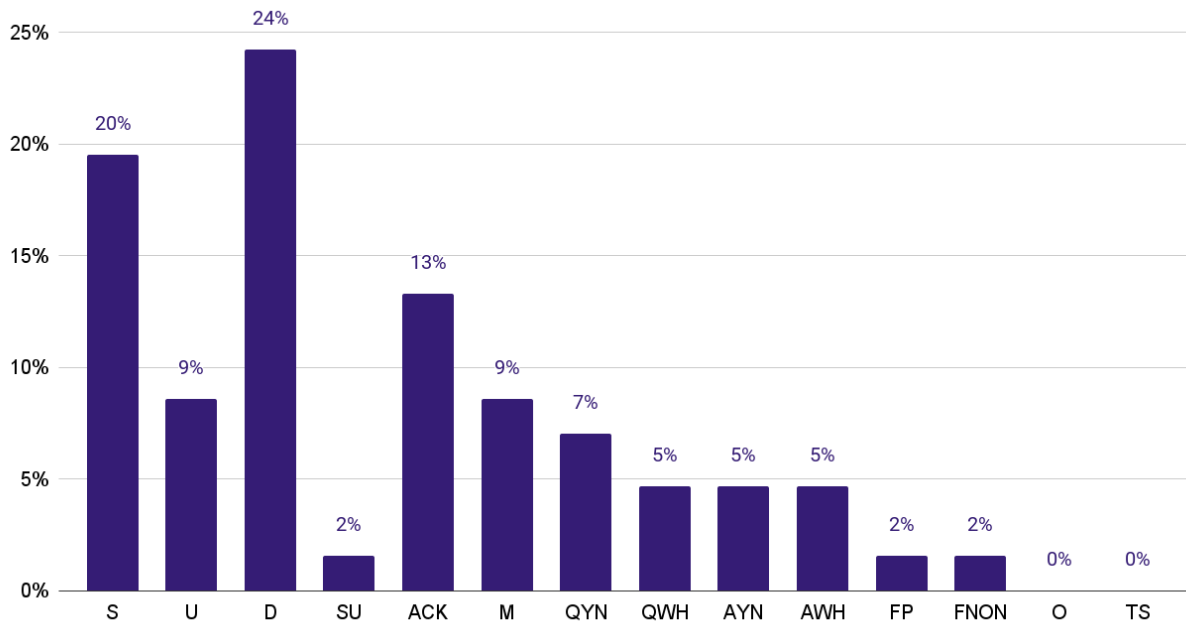


Figure 6.7: Percentage of tags during the activity completed with the Puzzle pair programming method

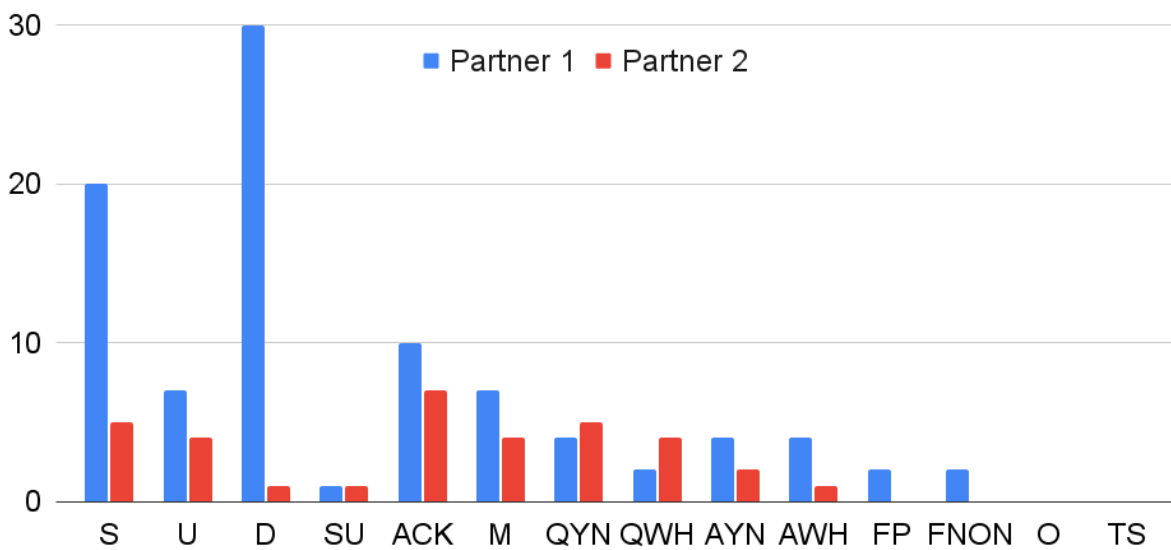


Figure 6.8: Count of tags for each participant during the activity completed with the Puzzle pair programming method

CHAPTER

7

CONCLUSIONS

7.1 Review

In Chapter 7, I provide a summary of how the research conducted in the previous chapters has helped validate the hypotheses and address the research questions stated in the introduction. To reiterate, the research questions and hypotheses are as follows:

7.2 Research Questions

- RQ1 How do students collaborate when using the Puzzle method?
- RQ2 How are student perceptions about collaboration and computer science affected by when using the Puzzle method compared to traditional pair programming?
- RQ3 How do student behaviors differ when students use the traditional pair programming method compared to the Puzzle method?

7.3 Hypotheses

- HY1 Participants will have a more even distribution of talk because tasks will be evenly distributed to each participant.
- HY2 When using the Puzzle method participants will be more engaged with each other when communicating verbally through questions and giving responses.
- HY3 Very little student positioning will occur between students because one student will be unable to "take over" the program.
- HY4 Participants who complete activities using the Puzzle method will have statistically significant changes in their attitudes towards computer science.
- HY5 Participants will enjoy the Puzzle method more than traditional pair programming because it gives their roles during collaboration more structure and easier communication.
- HY6 Participants who enjoyed working with their partner will have positive attitudes towards the pair programming method they used.
- HY7 When using the Puzzle method, participants will have a more equal distribution of talk than when using the traditional method.
- HY8 When using the Puzzle method, participants will use similar types of language to talk to each other (such as questions and answers) than when using the traditional method.
- HY9 When using the Puzzle method, less student positioning will occur than when using the traditional method.
- HY10 Students will prefer the Puzzle method over the traditional method because it encourages engagement with their partner and is an overall more fun and enjoyable way to collaborate.

7.4 Summary of Results

Research Question 1

Research question 1 is "How do students collaborate when using the Puzzle method?". After reviewing previous work investigating pair programming with K-12 students, I learned that the results of these works were inconsistent and showed that pair programming can create inequitable learning environments. From these realizations, I chose to investigate student

behavior while completing programming activities with the Puzzle pair programming method. Puzzle had been used in a previous study that showed that students enjoyed using it the most compared to other variations of pair programming. From this study I learned that when collaborating, student's distribution of talk may not give a clear picture of the behavior between students and that student positioning can occur even in pairs that had participated equally in conversations when collaborating. I was also able to determine a pattern of dialogue for pairs that collaborated more effectively, which consisted of asking questions and receiving answers rather than talking in statements. This study gave us insight into how students collaborate using these types of structured roles for the first time.

Research Question 2

Research question 2 is "How are student perceptions about collaboration and computer science affected by when using the Puzzle method compared to traditional pair programming?". After conducting a study investigating the effect of traditional and Puzzle pair programming on student attitudes toward collaboration and computer science, I learned that student preferences may affect their perceptions of each collaboration method. When examining the changes in student attitudes, we found that there were positive and negative changes in the Puzzle and traditional participants groups. However, we also found that slightly more participants preferred the Puzzle method than the participants who preferred the traditional method. Finally, we found that the Puzzle method may encourage students to place a higher value on the outcomes of their collaboration instead of their collaboration behavior when self-evaluating success in a pairing. This study gave us insight into implications for using Puzzle in the classroom and how student attitudes should be considered in future research investigating pair programming with high school-age students.

Research Question 3

Research question 3 is "How do student behaviors differ when students use the traditional pair programming method compared to the Puzzle method?". Considering the findings found in Chapters 4 and Chapter 5, I conducted a study investigating student behaviors and perceptions of collaboration after completing programming activities using both traditional and Puzzle pair programming methods. I learned that students tend to use a broader range of language when communicating using the Puzzle method than traditional pair programming. Another interesting finding from this study is the participant who was the navigator preferred Puzzle, while the partner who was the driver, preferred the driver-navigator method. This study gave us insight into why students may prefer one method over the other but also helped us develop

future research directed that should be conducted to holistically investigate the Puzzle method and its affects.

Hypothesis 1: False

Hypothesis HY 1 is that "Participants will have a more even distribution of talk because tasks will be evenly distributed to each participant.". HY 1 is investigated by work in Chapter 4 where I examined the distribution of talk of four pairs of students who completed activities using the Puzzle method. **From this study we found that the overall distribution of talk of participants was not equal between any of the pairs in our study.** This was surprising since we found one of the pairs to behave more collaboratively. Furthermore, this led us to question what might be another method of determining the distribution of talk between students. It was determined that assigning different weights to different parts of the participants' dialogue in the calculation of time might be more insightful. This is because certain sections of the conversation may be of greater significance in terms of the actions carried out by the participants.

Hypothesis 2: True

Hypothesis HY 2 is that "When using the Puzzle method participants will be more engaged with each other when communicating verbally through questions and giving responses.". HY 2 is supported by work in Chapter 4 where I examined the content of talk of four pairs of students completing activities using the Puzzle method. **Findings from this study suggest that there tends to be a pattern of dialog interactions between pairs that are less collaborative versus more collaborative.** Pairs that were less collaborative had a dynamic where the more dominant participant said more statements, while the other **participant said acknowledgments or asked questions.** In these instances, the dominant participant also said a great amount of *to self* statements. The pair who had more collaborative behaviors, pair 3, communicated mostly through yes/no and wh-questions and statements. In addition, in pair 3, we saw the least number of *to self* tags from both participants. Therefore, we suggest that students use practices similar to interviewee think-aloud protocols that are designed to help others understand the thought processes of each person in the pairing. This will ensure that students communicate more openly about their plans and encourage collaborative behavior.

Hypothesis 3: False

Hypothesis HY 3 is that "Very little student positioning will occur between students because one student will be unable to "take over" the program.". HY 3 is disproved by work in Chapter 4 where I examined the student positioning of four pairs of students completing activities

using the Puzzle method. **The findings from Chapter 4 show that the Puzzle method may not deter student attempts to position themselves as having a greater intellectual ability than their peer.** Of the four pairs in this study, there was evidence of student positioning in three. This could be due to previous relationships established through the internship program from which the participants were recruited. The most interesting finding was that the pair that did not have student positioning began their collaboration by further establishing their roles when starting the activity. We believe that this was the key to this pair's ability to collaborate successfully. In contrast, pairs who did not define their communication style or needs had less collaborative communication and mostly talked to themselves through problems instead of engaging their partner. CS pedagogy encourages students to plan their project development, however, this evidence supports the need to also plan collaboration practices, which may be especially useful for geographically separated or asynchronous teams.

Hypothesis 4: Inconclusive

Hypothesis HY 4 is that "Participants who complete activities using the Puzzle method will have statistically significant changes in their attitudes towards computer science.". HY 4 is supported by work in Chapter 5 where I examined the effect of traditional and Puzzle programming on student attitudes. **The findings of this study indicate that there were no significant differences in participant responses from the pre and post survey or when comparing post survey scores of the Puzzle and traditional groups. However, we did find that there was a positive attitude shift for students in both groups relating to attitudes toward collaboration and computing confidence.** The two questions that showed increased confidence were, (Q13) " I consider myself to be a computer programmer." and (Q14) "I take pride in my computing abilities." The change in confidence for these specific questions are interesting because they are also related to student identity in computer science. This can foster a sense of belonging and community, which in turn could lead to interest or retention of students in CS. I mark this HY 4 inconclusive due to the limitations of the study, as more data is needed.

Hypothesis 5: True

Hypothesis HY 5 is that "Participants will enjoy the Puzzle method more than traditional pair programming because it gives their roles during collaboration more structure and easier communication. ". HY 5 is supported by work in Chapter 5 where I examined the effect of traditional and Puzzle programming on student attitudes and perceptions. **The results indicate that there were more students who preferred the Puzzle method compared to the students who preferred the traditional method.** When we looked closer at why this was occurring, it

seemed that some students liked the level of collaboration that Puzzle encouraged. However, there were also negative views on the subject. We saw that students found Puzzle restrictive while traditional let them be more creative. The most intriguing finding from this analysis is that the purpose of Puzzle was fulfilled in forcing students to communicate more when collaborating, but not all students liked how much communication was required to be successful. Therefore, in line with the findings in Chapter 4, we suggest that Puzzle be used as a training tool to teach best practices to students when beginning to learn about collaboration. Then, after some familiarity with how to best collaborate with a partner, allow the students to choose their preferred method.

Hypothesis 6: Inconclusive

Hypothesis HY 6 is that "Participants who enjoyed working with their partner will have positive attitudes towards the pair programming method they used. ". HY 6 is supported by work in Chapter 5 where I examined the effect of traditional and Puzzle programming on student attitudes and perceptions. **The findings of Chapter 5 indicate that there was little relationship between the participants' perceptions of their partner and their perceptions of each collaboration method. However, we found that there were similarities and differences in perceptions of collaboration between the two groups.** When discussing their own collaboration, participants in both groups used their accomplishments and completed activities as metrics of collaborativeness. Similarly, when discussing their partners' behaviors, participants in the Puzzle group believed that their partners were working collaboratively based on their outcomes, not on their behaviors. However, participants in the traditional group considered communication and understanding as attributes of successful collaboration of their partners. These findings suggest that the Puzzle method may cause students to place a higher value on producing and accomplishing their end goal while the traditional method encourages students to place value on communication. Although the overall goal of the Puzzle method is to encourage students to communicate while collaborating.

Hypothesis 7: Inconclusive

Hypothesis HY 7 is that "When using the Puzzle method, participants will have a more equal distribution of talk than when using the traditional method. ". HY 7 is supported by work in Chapter 6 where I investigate the differences in student behavior when completing programming activities with the traditional and Puzzle pair programming method. We also interviewed the students to determine their perceptions of each method. **The results indicate that students had unequal distribution of talk during when using the Puzzle method. Additionally,**

when using the traditional method, the navigator spoke the most, which is expected for this collaboration method. When looking through the lens of the student responses to the interview questions, we can see that Partner 2 preferred Puzzle, but thought the communication was "hard". This may indicate why he did not contribute as much as Partner 1. This also may indicate that Partner 2 placed a higher value on being able to have hands-on experiences compared to 'more effective' communication. HY 7 is marked as inconclusive due to the small sample size and lack of role change in the traditional pair programming method.

Hypothesis 8: True

Hypothesis HY 8 is that "When using the Puzzle method, participants will use similar types of language to talk to each other (such as questions and answers) than when using the traditional method." HY 8 is supported by work in Chapter 6 where I investigate the differences in student behavior when completing programming activities with the traditional and Puzzle pair programming method. We also interviewed students to determine their perceptions of each method. **The findings of this study indicate that Partner 1 and Partner 2 behaved more collaboratively using the Puzzle method.** There was an overall increase in the types of conversations they used to engage each other which included questions, meta comments, and acknowledgments when using the Puzzle method. This may have been caused by the lack of communication restrictions Puzzle puts on students. The traditional method specifically states that the navigator is to give instructions to the driver, which in turn, caused the pair to communicate using mostly explicit instructions. However, the Puzzle method may encourage students to ask more questions and communicate with each other more like a conversation.

Hypothesis 9: Inconclusive

Hypothesis HY 9 is that "When using the Puzzle method, less student positioning will occur than when using the traditional method." HY 9 is supported by work in Chapter 6 where I investigate the differences in student behavior when completing programming activities with the traditional and Puzzle pair programming method. We also interviewed students to determine their perceptions of each method. **The findings indicate that dominance in each of the collaboration methods still remains even when partners are familiar with each other.** During the traditional pair programming, Partner 2 attempted to position himself as more intelligence by dismissing Partner 1's concerns. However, during the Puzzle activity, Partner 1 took on a more dominant role by being the one to read the instructions and tell Partner 2 what blocks were needed to complete the program.

Hypothesis 10: Inconclusive

Hypothesis HY 10 is that "Students will prefer the Puzzle method over the traditional method because it encourages engagement with their partner and is an overall more fun and enjoyable way to collaborate. ". HY 10 is supported by work in Chapter 6 where I investigate the differences in student behavior when completing programming activities with the traditional and Puzzle pair programming method. We also interviewed students to determine their perceptions of each method. **The findings of Chapter 6 indicate that the use of different collaboration methods in programming activities can lead to differences in student preferences and perceptions. However, still lead students to value the importance of collaboration and teamwork in achieving the end goal.** Partner 1 preferred using the traditional method because of the clear instruction provided by Partner 2, the navigator, and the hands-on experience as the driver. While Partner 2 preferred the Puzzle method also because of its hands-on learning for each participant and the engagement between himself and Partner 1. The preferences of Partner 1 and Partner 2 after experiencing both methods provide evidence that Puzzle may be a more helpful collaboration tool in the classroom, since both students would be able to gain hands-on experience compared to the traditional method, which does not always grant each student an equal amount of time as the driver. In previous work, researchers have shown that the navigator role does not provide the same benefits as the driver role. These findings support this work in showing that the role of navigator may not be the best way for some students to learn collaborative skills when programming.

7.5 Contributions

In conclusion, this research presents the following contributions:

1. I conducted the first study investigating student behavior while using a collaboration method that uses block palettes in a block-based programming environment to divide tasks among pairs when pair programming. From this study, we were able to gain insight into collaborative behavior patterns, such as question-answer, that may indicate more equitable collaboration between partners.
2. I conducted the first study to investigate student attitudes, collaboration preferences, and perceptions of their partner after using traditional and Puzzle pair programming. From this study, we were able to show that the Puzzle method may have a positive affect on student attitudes toward collaboration and computing confidence. We were also able to show that students felt they communicated more using Puzzle although, some students did not enjoy it.

3. I conducted the first study to compare student behavior when using the traditional Driver-Navigator method between behavior when using the Puzzle method. From this study, we were able to gain insight into how the same students collaborate in each method. We found that when students collaborate using the Puzzle method, they use a wider range of language to communicate with each other, such as asking questions or sharing meta-comments, than when using traditional pair programming.
4. I modified a hybrid conversational analysis framework that combined the analysis of student talk distribution, content of talk, and student positioning with an established *a priori* codebook used for discourse analysis of pairs. This allows the computing education research community to have a consistent framework to evaluate student behavior in pair programming studies and compare results to other work.
5. I provided the computing education research community with suggestions of how to better investigate pair programming with secondary students and how to use block palettes in a block-based programming environment to divide tasks among pairs when pair programming. I also provided the K-12 computer science education community with implications of using Puzzle pair programming in the classroom which include, using Puzzle as a learning tool before pair programming and to consider student's preferences of communication when creating pairs for pair programming.

7.6 Future Work

Considerably more work is needed to understand the lasting effects of the Puzzle method on students' experiences and perceptions of computer science. A natural progression of this research involves comparing student behaviors in traditional pair programming and the Puzzle method with more participants. Additionally, future studies should explore altering the order of collaboration methods, such as starting with Puzzle before traditional pair programming, and investigating student engagement through methods like role and partner switching. Implementing replication studies with larger participant numbers in formal classroom environments can provide further context on how partner relationships influence collaboration with the Puzzle method, and how its use affects students' attitudes toward CS and programming. It's also important for future research to assess student learning gains and self-efficacy when using the traditional method compared to the Puzzle method.

REFERENCES

- Daniel Felipe Almanza Cortés, Manuel Felipe Del Toro Salazar, Ricardo Andrés Urrego Arias, Pedro Guillermo Feijoo Garcia, and Fernando De la Rosa. Scaffolded block-based instructional tool for linear data structures: A constructivist design to ease data structures' understanding. 2019.
- B Barron, K Gomez, N Pinkard, and CK Martin. The digital youth network: Cultivating digital citizenship in urban communities, 2014.
- Brigid Barron. When smart groups fail. *The journal of the learning sciences*, 12(3):307–359, 2003.
- Liat Bodaker and Rinat B Rosenberg-Kima. Online pair-programming: Elementary school children learning scratch together online. *Journal of Research on Technology in Education*, pages 1–18, 2022.
- Corey Brady, Brian Broll, Gordon Stein, Devin Jean, Shuchi Grover, Veronica Cateté, Tiffany Barnes, and Ákos Lédeczi. Block-based abstractions and expansive services to make advanced computing concepts accessible to novices. *Journal of Computer Languages*, page 101156, 2022.
- Grant Braught, Tim Wahls, and L Marlin Eby. The case for pair programming in the computer science classroom. *ACM Transactions on Computing Education (TOCE)*, 11(1):1–21, 2011.
- Brian Broll, Péter Völgyesi, János Sallai, and Akos Lédeczi. Netsblox: A visual language and web-based environment for teaching distributed programming, 2016.
- Brian Broll, Akos Lédeczi, Peter Volgyesi, Janos Sallai, Miklos Maroti, Alexia Carrillo, Stephanie L Weeden-Wright, Chris Vanags, Joshua D Swartz, and Melvin Lu. A visual programming environment for learning distributed programming. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*, pages 81–86, 2017.
- Brian Broll, Akos Lédeczi, Gordon Stein, Devin Jean, Corey Brady, Shuchi Grover, Veronica Catete, and Tiffany Barnes. Removing the walls around visual educational programming environments. In *2021 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, pages 1–9. IEEE, 2021.
- William M Bukowski, Betsy Hoza, and Michel Boivin. Measuring friendship quality during pre-and early adolescence: The development and psychometric properties of the friendship qualities scale. *Journal of social and Personal Relationships*, 11(3):471–484, 1994.
- Jennifer Burg, V Paúl Pauca, William Turkett, Errin Fulp, Samuel S Cho, Peter Santiago, Daniel Cañas, and H Donald Gage. Engaging non-traditional students in computer science through socially-inspired learning and sustained mentoring. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, pages 639–644, 2015.

- Ünal Çakıroğlu, İsak Çevik, Engin Köşeli, and Merve Aydın. Understanding students' abstractions in block-based programming environments: A performance based evaluation. *Thinking Skills and Creativity*, 41:100888, 2021.
- Shannon Campe, Jill Denner, Emily Green, and David Torres. Pair programming in middle school: variations in interactions and behaviors. *Computer Science Education*, 30(1):22–46, 2020.
- Brenda M Capobianco. Undergraduate women engineering their professional identities. *Journal of Women and minorities in Science and Engineering*, 12(2-3), 2006.
- Veronica Catete, Amy Isvik, and Marnie Hill. A framework for socially-relevant service-learning internship experiences for high school students. In *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education-Volume 1*, pages 815–821, 2022.
- Funda Dağ. Prepare pre-service teachers to teach computer programming skills at k-12 level: Experiences in a course. *Journal of Computers in Education*, 6(2):277–313, 2019.
- Maggie Dahn and David DeLiema. Dynamics of emotion, problem solving, and identity: Portraits of three girl coders. *Computer Science Education*, 30(3):362–389, 2020.
- Jill Denner, Linda Werner, Shannon Campe, and Eloy Ortiz. Pair programming: Under what conditions is it advantageous for middle school students? *Journal of Research on Technology in Education*, 46(3):277–296, 2014.
- Jill Denner, Emily Green, and Shannon Campe. Learning to program in middle school: How pair programming helps and hinders intrepid exploration. *Journal of the Learning Sciences*, 30(4-5):611–645, 2021.
- Remy Dou, Karina Bhutta, Monique Ross, Laird Kramer, and Vishodana Thamocharan. The effects of computer science stereotypes and interest on middle school boys' career intentions. *ACM Transactions on Computing Education (TOCE)*, 20(3):1–15, 2020.
- Wendy DuBow, Alexis Kaminsky, and Joanna Weidler-Lewis. Multiple factors converge to influence women's persistence in computing: A qualitative analysis. *Computing in Science & Engineering*, 19(3):30–39, 2017.
- Michael P Fay and Michael A Proschan. Wilcoxon-mann-whitney or t-test? on assumptions for hypothesis tests and multiple interpretations of decision rules. *Statistics surveys*, 4:1, 2010.
- National Center for Women & Information Technology. Ncwit pair programming student final assessment. <https://www.ncwit.org/file/pair-programming-student-final-assessment>, 2016.
- Cory Gleasman and ChanMin Kim. Pre-service teacher's use of block-based programming and computational thinking to teach elementary mathematics. *Digital Experiences in Mathematics Education*, 6:52–90, 2020.

- Jeff Gray, Kathy Haynie, Fran Trees, Owen Astrachan, Chinma Uche, Siobhan Cooney, and Richard Kick. Infusing cooperative learning into ap computer science principles courses to promote engagement and diversity. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, pages 1190–1196, 2019.
- Brian Hanks, Sue Fitzgerald, Renée McCauley, Laurie Murphy, and Carol Zander. Pair programming in education: A literature review. *Computer Science Education*, 21(2):135–173, 2011.
- Brian Harvey, Daniel D Garcia, Tiffany Barnes, Nathaniel Titterton, Daniel Armendariz, Luke Segars, Eugene Lemon, Sean Morris, and Josh Paley. Snap!(build your own blocks). In *Proceeding of the 44th ACM technical symposium on Computer science education*, pages 759–759, 2013.
- Myles Hollander, Douglas A Wolfe, and Eric Chicken. *Nonparametric statistical methods*. John Wiley & Sons, 2013.
- Elizabeth V Howard, Donna Evans, Jill Courte, and Cathy Bishop-Clark. A qualitative look at alice and pair-programming. In *Proceedings of ISECON 2006*, 2006.
- Aleata Hubbard, Yvonne Kao, and Danielle Brown. Designing think-aloud interviews to elicit evidence of computer science pedagogical content knowledge. *AERA*, 2016.
- Ian Hutchby and Robin Wooffitt. *Conversation analysis*. Polity, 2008.
- Dan Ingalls, Ted Kaehler, John Maloney, Scott Wallace, and Alan Kay. Back to the future: the story of squeak, a practical smalltalk written in itself. In *Proceedings of the 12th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications*, pages 318–326, 1997.
- Olivera Iskrenovic-Momcilovic. Pair programming with scratch. *Education and Information Technologies*, 24:2943–2952, 2019.
- Amy Isvik, Veronica Cateté, Dave Bell, Isabella Gransbury, and Tiffany Barnes. Infusing computing: Moving a service oriented internship program online. In *2021 Conference on Research in Equitable and Sustained Participation in Engineering, Computing, and Technology (RESPECT)*, pages 1–5. IEEE, 2021.
- Alan Kay, Kim Rose, Dan Ingalls, Ted Kaehler, John Maloney, Scott Wallace, et al. Etoys & simstories. *ImagiLearning Internal Document*, 1997.
- Colleen M Lewis and Niral Shah. How equity and inequity can emerge in pair programming. In *Proceedings of the eleventh annual international conference on international computing education research*, pages 41–50, 2015.
- Janet Liebenberg, Elsa Mentz, and Betty Breed. Pair programming and secondary school girls' enjoyment of programming and the subject information technology (it). *Computer Science Education*, 22(3):219–236, 2012.

- Yuhan Lin and David Weintrop. The landscape of block-based programming: Characteristics of block-based environments and how they support the transition to text-based programming. *Journal of Computer Languages*, 67:101075, 2021.
- Nicholas Lytle, Alexandra Milliken, Veronica Cateté, and Tiffany Barnes. Investigating different assignment designs to promote collaboration in block-based environments. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*, pages 832–838, 2020.
- Moira Maguire and Brid Delahunt. Doing a thematic analysis: A practical, step-by-step guide for learning and teaching scholars. *All Ireland Journal of Higher Education*, 9(3), 2017.
- Phil Maguire, Rebecca Maguire, Philip Hyland, and Patrick Marshall. Enhancing collaborative learning using pair programming: Who benefits? *AISHE-J: The All Ireland Journal of Teaching and Learning in Higher Education*, 6(2), 2014.
- John Maloney, Leo Burd, Yasmin Kafai, Natalie Rusk, Brian Silverman, and Mitchel Resnick. Scratch: a sneak preview [education]. In *Proceedings. Second International Conference on Creating, Connecting and Collaborating through Computing, 2004.*, pages 104–109. IEEE, 2004.
- Charlie McDowell, Linda Werner, Heather Bullock, and Julian Fernald. The effects of pair-programming on performance in an introductory programming course. In *Proceedings of the 33rd SIGCSE technical symposium on Computer science education*, pages 38–42, 2002.
- Charlie McDowell, Linda Werner, Heather E Bullock, and Julian Fernald. Pair programming improves student retention, confidence, and program quality. *Communications of the ACM*, 49(8):90–95, 2006.
- Dawn McKinney and Leo F Denton. Developing collaborative skills early in the cs curriculum in a laboratory environment. In *Proceedings of the 37th SIGCSE technical symposium on Computer science education*, pages 138–142, 2006.
- Emilia Mendes, Lubna Basil Al-Fakhri, and Andrew Luxton-Reilly. Investigating pair-programming in a 2nd-year software development and design computer science course. In *Proceedings of the 10th annual SIGCSE conference on Innovation and technology in computer science education*, pages 296–300, 2005.
- Janvi Nandwani. The creation, use, and impact of block-based programming curriculum. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 2*, pages 1238–1238, 2022.
- Stamatios Papadakis. Is pair programming more effective than solo programming for secondary education novice programmers?: A case study. *International Journal of Web-Based Learning and Teaching Technologies (IJWLTT)*, 13(1):1–16, 2018.
- Piumi Perera, Geethya Tennakoon, Supunmali Ahangama, Rangana Panditharathna, and Buddhika Chathuranga. A systematic mapping of introductory programming languages for novice learners. *IEEE Access*, 9:88121–88136, 2021.

- Fernando J Rodríguez, Kimberly Michelle Price, and Kristy Elizabeth Boyer. Exploring the pair programming process: Characteristics of effective collaboration. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*, pages 507–512, 2017.
- Omar Ruvalcaba, Linda Werner, and Jill Denner. Observations of pair programming: Variations in collaboration across demographic groups. In *Proceedings of the 47th ACM technical symposium on computing science education*, pages 90–95, 2016.
- Norsaremah Salleh, Emilia Mendes, and John Grundy. Empirical studies of pair programming for cs/se teaching in higher education: A systematic literature review. *IEEE Transactions on Software Engineering*, 37(4):509–525, 2010.
- Niral Shah, Colleen Lewis, and Roxane Caires. Analyzing equity in collaborative learning situations: A comparative case study in elementary computer science. Boulder, CO: International Society of the Learning Sciences, 2014.
- Samuel Sanford Shapiro and Martin B Wilk. An analysis of variance test for normality (complete samples). *Biometrika*, 52(3/4):591–611, 1965.
- Andy Smith, Bradford Mott, Sandra Taylor, Aleata Hubbard-Cheououa, James Minogue, Kevin Oliver, and Cathy Ringstaff. Toward a block-based programming approach to interactive storytelling for upper elementary students. In *Interactive Storytelling: 13th International Conference on Interactive Digital Storytelling, ICIDS 2020, Bournemouth, UK, November 3–6, 2020, Proceedings 13*, pages 111–119. Springer, 2020.
- Andreas Stolcke, Klaus Ries, Noah Coccaro, Elizabeth Shriberg, Rebecca Bates, Daniel Jurafsky, Paul Taylor, Rachel Martin, Carol Van Ess-Dykema, and Marie Meteer. Dialogue act modeling for automatic tagging and recognition of conversational speech. *Computational linguistics*, 26(3):339–373, 2000.
- Jennifer Tsan, Kristy Elizabeth Boyer, and Collin F Lynch. How early does the cs gender gap emerge? a study of collaborative problem solving in 5th grade computer science. In *Proceedings of the 47th ACM technical symposium on computing science education*, pages 388–393, 2016.
- Jennifer Tsan, Jessica Vandenberg, Zarifa Zakaria, Joseph B Wiggins, Alexander R Webber, Amanda Bradbury, Collin Lynch, Eric Wiebe, and Kristy Elizabeth Boyer. A comparison of two pair programming configurations for upper elementary students. In *Proceedings of the 51st ACM technical symposium on computer science education*, pages 346–352, 2020.
- Man-Yin Tsang, Lisa Tutty, and Carl-Georg Bank. The effectiveness of cooperative learning in teaching quantitative reasoning with ternary diagrams in a science class. *Journal of College Science Teaching*, 52(7):111–118, 2023.
- Karthikeyan Umapathy and Albert D Ritzhaupt. A meta-analysis of pair-programming in computer programming courses: Implications for educational practice. *ACM Transactions on Computing Education (TOCE)*, 17(4):1–13, 2017.

- Alexandria Katarina Vail and Kristy Elizabeth Boyer. Identifying effective moves in tutoring: On the refinement of dialogue act annotation schemes. In *Intelligent Tutoring Systems: 12th International Conference, ITS 2014, Honolulu, HI, USA, June 5-9, 2014. Proceedings 12*, pages 199–209. Springer, 2014.
- Lucas Vasconcelos and ChanMin Kim. Preparing preservice teachers to use block-based coding in scientific modeling lessons. *Instructional science*, 48(6):765–797, 2020.
- Lev Vygotsky et al. *Interaction between learning and development*. Linköpings universitet, 2011.
- Xuefeng Wei, Lin Lin, Nanxi Meng, Wei Tan, Siu-Cheung Kong, et al. The effectiveness of partial pair programming on elementary school students’ computational thinking skills and self-efficacy. *Computers & education*, 160:104023, 2021.
- David Weintrop. Block-based programming in computer science education. *Communications of the ACM*, 62(8):22–25, 2019.
- David Weintrop and Uri Wilensky. Comparing block-based and text-based programming in high school computer science classrooms. *ACM Transactions on Computing Education (TOCE)*, 18(1):1–25, 2017.
- Linda Werner and Jill Denning. Pair programming in middle school: What does it look like? *Journal of Research on Technology in Education*, 42(1):29–49, 2009.
- Linda Werner, Jill Denner, Michelle Bliesner, and Pat Rex. Can middle-schoolers use storytelling alice to make games? results of a pilot study. In *Proceedings of the 4th International Conference on foundations of digital games*, pages 207–214, 2009.
- Linda L Werner, Brian Hanks, and Charlie McDowell. Pair-programming helps female computer science students. *Journal on Educational Resources in Computing (JERIC)*, 4(1):4–es, 2004.
- Timothy J Weston, Wendy M Dubow, and Alexis Kaminsky. Predicting women’s persistence in computer science-and technology-related majors from high school to college. *ACM Transactions on Computing Education (TOCE)*, 20(1):1–16, 2019.
- Laurie Williams. Integrating pair programming into a software development process. In *Proceedings 14th Conference on Software Engineering Education and Training. In search of a software engineering profession* (Cat. No. PR01059), pages 27–36. IEEE, 2001.
- Laurie Williams, Robert R Kessler, Ward Cunningham, and Ron Jeffries. Strengthening the case for pair programming. *IEEE software*, 17(4):19–25, 2000.
- Hui Yang, Diane Coddling, Chrystalla Mouza, and Lori Pollock. Broadening participation in computing: Promoting affective and cognitive learning in informal spaces. *TechTrends*, 65: 196–212, 2021.
- Zarifa Zakaria. *Collaborative Programming Practices in Elementary Classrooms: Exploring the Roles of Gender, Motivation, and Collaborative Discourse*. North Carolina State University, 2021.

Zarifa Zakaria, Jessica Vandenberg, Jennifer Tsan, Danielle Cadieux Boulden, Collin F Lynch, Kristy Elizabeth Boyer, and Eric N Wiebe. Two-computer pair programming: Exploring a feedback intervention to improve collaborative talk in elementary students. *Computer Science Education*, 32(1):3–29, 2022.

Baichang Zhong, Qiyun Wang, and Jie Chen. The impact of social factors on pair programming in a primary school. *Computers in Human Behavior*, 64:423–431, 2016.

APPENDICES

APPENDIX

A

SURVEY

The questions asked on the pre and post survey are documented in Table A.1

Table A.1: Questions asked in the pre and post survey, the constructs they measure, and their respective answer scales.

Pre and Post Survey		
Constructs	Questions	Scale
Attitude Toward Collaboration	I learn more when I work with others. I have more fun when I work with others. When more than one person works on a project, we have better ideas and make better things.	5
Social Support	Important people in my life think its a good thing for me to learn about technology. I believe people like me can do well learning technology. My family like me to learn about technology. Other students generally think its cool that I learn about technology.	4
Computing Confidence	I am good with computers. I feel confident about my ability to use computers. When I use technology I think about how it works. I like the challenge of learning how to use a new technology. I feel like I belong in computer science. I consider myself to be a computer programmer. I take pride in my computing abilities.	5
Programming Interest	How interested are you in programming computers or other technologies (in other words writing code)? How much do you want to learn how to program computers or other technologies? How much do you want to learn computer science?	4
Post Survey		
Partner Perceptions	I feel happy when I am with my partner. If my partner stopped coming to camp I would miss them. When I do a good job at something my partner is happy for me. Sometimes my partner does things for me or makes me feel special. I can count on my partner for help. I think about my partner even when my partner is not around. I consider my partner to be a friend. My partner and I spend time together outside of this class.	5
	Did you find the pair programming method to be helpful when completing the programming activities?	3
	Please explain why or why not	Open
	Do you feel you worked collaboratively with your partner?	2
	Please explain why or why not	Open

APPENDIX

B

INTERVIEW PROTOCOL

The interview questions asked in Chapter 6 are documented in Table B.1.

Table B.1: The questions asked in the participant interviews in Chapter 6

Collaboration Questions
How did you feel collaborating with the Driver-Navigator method?
What did you enjoy about the Driver-Navigator method?
What did you dislike about the Driver-Navigator method?
How did you feel about your communication with your partner while using the Driver-Navigator method?
How did you feel collaborating with the Puzzle method?
What did you enjoy about the Puzzle method?
What did you dislike about the Puzzle method?
How did you feel about your communication with your partner while using the Puzzle method?
Which collaboration method would you want to use again? Why?

APPENDIX

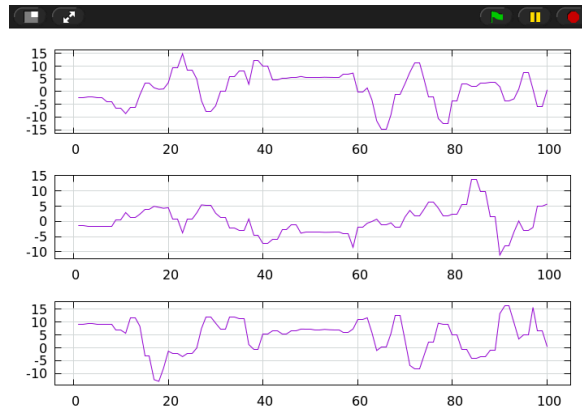
C

EXAMPLE PROGRAMMING ACTIVITY

The instructions of an example programming activity used in Chapter 4 and Chapter 6 is shown below.



Intro to PhonoIoT - Accelerometer Plotter



Summary

In this project, you will create a project which collects a stream of data from the accelerometer and plots the most-recent 10 seconds of data for each axis as a “sliding window” (a portion of time) graph.

Before You Begin

From the previous lessons, you should already be familiar with sensor streaming with PhonoIoT, at least with the accelerometer. If not, please review the previous lesson before starting.

You should also already be familiar with using the Chart service in NetsBlox from previous lessons with ThingSpeak. We will be doing very simple plotting, but if you find yourself struggling, look back at the code/materials from previous lessons.

Gathering Data

As in the previous project, we will start by defining our device variable, setting credentials, and listening to accelerometer updates every 100ms. We want to run this immediately when the project is started, so you can **put it in the stage** with a “When green flag clicked” block:



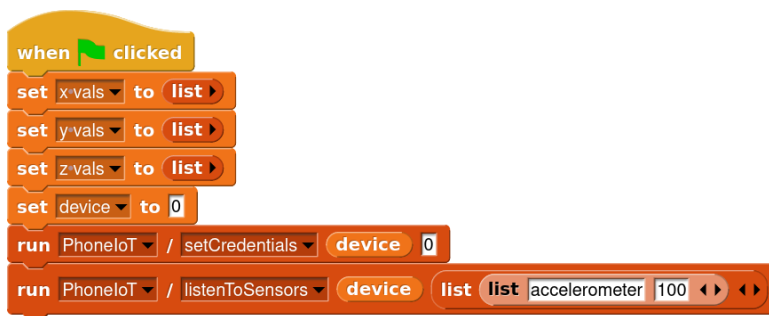
Next, go ahead and make your accelerometer message type (“Make a message type” near the bottom of the Network tab). We need to call the message type “accelerometer” and give it fields “x”, “y”, and “z”.



Next, grab a “When I receive” block from the Network tab and pick the accelerometer message type you just made from the dropdown.

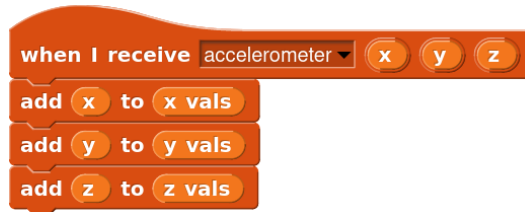


At this point, we can receive accelerometer updates, but we have nowhere to put them. We want to plot the x, y, and z values of the accelerometer, so we need three separate lists to hold the values. To do this, we’ll create three new variables, called “x vals”, “y vals”, and “z vals” and set them to an empty list when the project is started. Here’s the updated code for your “When green flag clicked” script:



Note: to get the empty list block, grab a “list” block from the Variables tab and click the black arrow pointing left to get rid of the single default item it tries to add.

Now, in the accelerometer message receiver, we can add the x, y, and z values to these lists:



```
when I receive accelerometer x y z
add x to x vals
add y to y vals
add z to z vals
```

Do you see any problem with this? Hint: how long will these lists get?

The current code will result in adding one value to each of our three lists every time we receive an update, which is 10 times per second. This means that the lists will become longer and longer until they are impractically large to work with (for plotting) and will slow down our program. Not to mention that the time axis of the plot will be squeezed smaller and smaller as more data comes in.

To solve this, we'll impose some hard limit on the number of points we can store. Let's say we only want to store the most-recent 10 seconds of data. Well, 10 samples per second over 10 seconds means our lists need to max out at around 100 items. When we try to add a new value to a list, we can check how many items are already present - if it's less than 100, we can just add the item directly. Otherwise (if it has 100 items), we need to remove the old (first) sample (making the list have 99 items) and add the new sample to the end. Here's some code that would accomplish this:



```
when I receive accelerometer x y z
if length of x vals > 99
delete 1 of x vals
add x to x vals
if length of y vals > 99
delete 1 of y vals
add y to y vals
if length of z vals > 99
delete 1 of z vals
add z to z vals
```

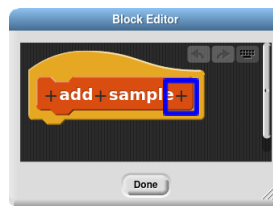
If you look closely, we just repeated effectively the same logic three times. It's never good to duplicate code like this. To fix this, we can create a custom block (function) that will perform this logic internally, and we only need to tell it what value to insert into what list.

If you scroll down to the bottom of the Variables tab, you'll see a button called "Make a block". Clicking that will bring you to the block creation window. Our new block will operate on lists, so select Lists from the tab selector (see below). In the text box below that, we put the name of the new block we want to make; we can call it "add sample". Leave "Command" selected. Then press OK.



After pressing OK, a window will pop up for you to edit your custom block's code. You can use the arrow-shaped handle at the bottom right of the window to make it larger if needed. If you accidentally close this window, you can go to the Custom tab, find your custom block, right click, and select "edit".

We will need to add two "parameters" (inputs) to our block. To add a parameter, click the "+" button on the right-most side of the main block in the block editor window (outlined below).

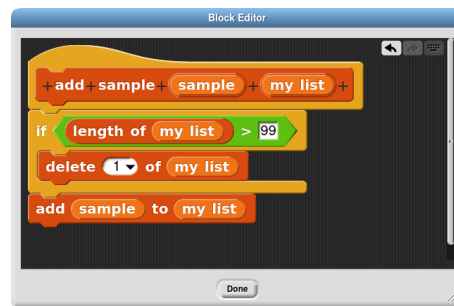


This will pop up a window to create the new input. Importantly, we must give it a name. For the first variable, we will call it "sample". *You can also select the type of input data you expect (number, list, etc.), but that is not necessary for now.* Select OK to create the input, then repeat

this process to make a second input called “my list”. Your editor should now look like the following:



Now we can add our code that the custom block should run. Remember that we already wrote the logic of this for x, y, and z separately - you can drag and drop in **one** of those copies. Importantly, instead of using “x”, “y”, or “z” specifically, we want to use “sample”, and instead of “x vals”, “y vals”, or “z vals” specifically, we want to use “my list”. *Note: You can drag and drop the “sample” and “my list” variable blocks from the main custom block in the block editor.* In the end, you should have something like the following:



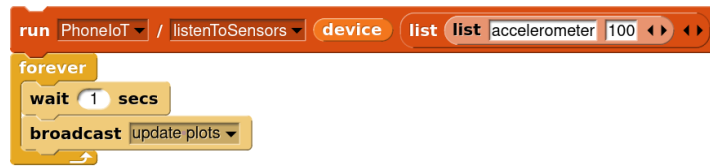
When you’re done editing the block, click “Done”. Now, we can use our custom block to replace all the nasty duplicated code we had before. You can find your new block in the Custom tab, and drag it into your code like any other block. Here is the new accelerometer receiver code:



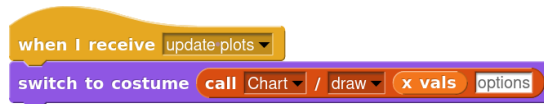
Note: For some code duplication this small, it might not seem worthwhile to make a custom block, but this is good practice. If you continue to develop your programming skills, eventually you will be writing very complex logic with hundreds of blocks/lines/commands, which you

absolutely cannot just copy and paste all over the place without making the whole project unmanageable to develop.

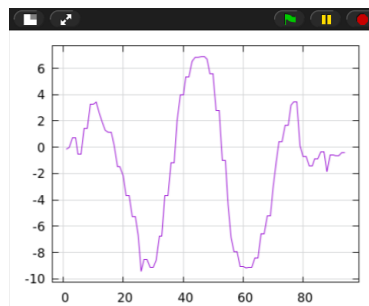
Now we have our lists of values, but we need to plot them and display them on the stage. We don't want to render new plots every time we get an accelerometer update, as that would be too slow. Instead, we'll create a forever loop in which we broadcast a new event called "update plots" every second. We can put this forever loop right after our listenToSensors run block. Here's a partial code snippet of the updated green flag clicked script:



Now **head over to the Sprite script** and grab a "When I receive" block from the Control tab to receive the "update plots" event. This sprite will be in charge of plotting the x values. When we receive the "update plots" event, we will render a new plot image with the Chart service and display it by setting it as our costume. *You should already be familiar with using the Chart service from previous lessons, but its usage is not overly complicated for our purposes (see below).*



If you run the current project (*make sure your phone is still awake and connected*), you should see something like the following:

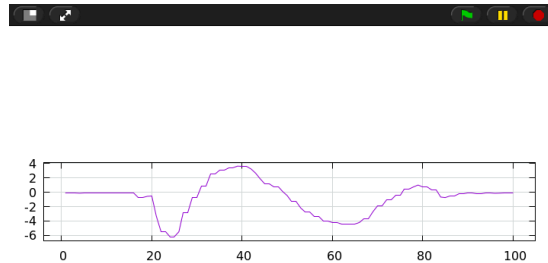


But we want to draw three plots, so we need to change the size of the image to be wide and short to fit three images vertically. Specifically, we want the image to take up the whole width,

but only a third of the height of the stage. To do this, we can pass the “width” and “height” optional parameters to the draw RPC:



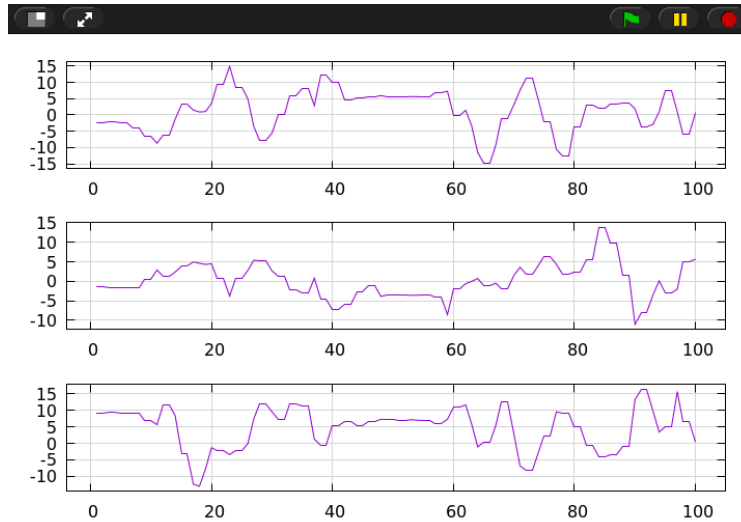
If you click this code to run it (or run the whole project), you'll notice that the image looks bad... This is because the default stage resolution is fairly small. To fix this, click the cog wheel near the top left of the NetsBlox project editor and select “Stage size”. We'll set this to a more practical resolution like 720x480. *When you resize the stage, you might need to drag the sprite back to the center.* Now you should see a clean plot (if you re-run the code):



From here, we'll create two more duplicate sprites. You can right click on your current sprite in the sprite list (bottom right panel) and select “duplicate” to create a copy of the sprite that has all the same code copied over. *The sprites will start out overlapping after being duplicated, so you'll need to drag them apart.*

These two new sprites will be for plotting the y and z values. Figure out which sprite is which by double clicking their entry in the sprite list (it will highlight them for you). We want the x sprite to be on the top, followed by the y and then the z sprite. *If you like, you can rename them by selecting one and editing its name in a textbox near the top left of the screen.*

Now just change the code in the y and z sprites to instead draw the “y vals” and “z vals” (instead of “x vals” like it currently has copied from the first sprite). The final result should look something like the following:



And now you have a completed accelerometer plotter project!

Finished project:

<https://editor.netsblox.org/?action=present&Username=csfrontiers&ProjectName=accel-plotter-simle&editMode&noRun>

Ideas for modifications/improvements:

You may notice that the plots are somewhat spiky and can be subject to abrupt large changes in value. What if we wanted to somehow smooth these values out? Can you think of any mathematical tools that might be useful? Or even better, how you might do it here?

If you look at the x axis of our final plots, you'll see that the numbers range from 0 to 100. This is because we only gave "y" values (as in for plotting, not "y" from the accelerometer), so it just used 1,2,3,... for the x values by default. Can you modify your code to use time instead? Specifically, we want it to start at zero when the project starts, and increase over time, so after 20 seconds, it should read 20.0–30.0 (approximate) since we display (around) 10 seconds of data in each plot.

APPENDIX

D

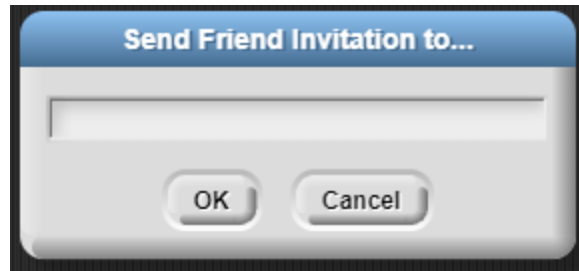
ACCESSING THE PUZZLE METHOD IN NETSBLOX

Instructions on how to access the Puzzle method in the block-based programming environment NetsBlox are shown below:

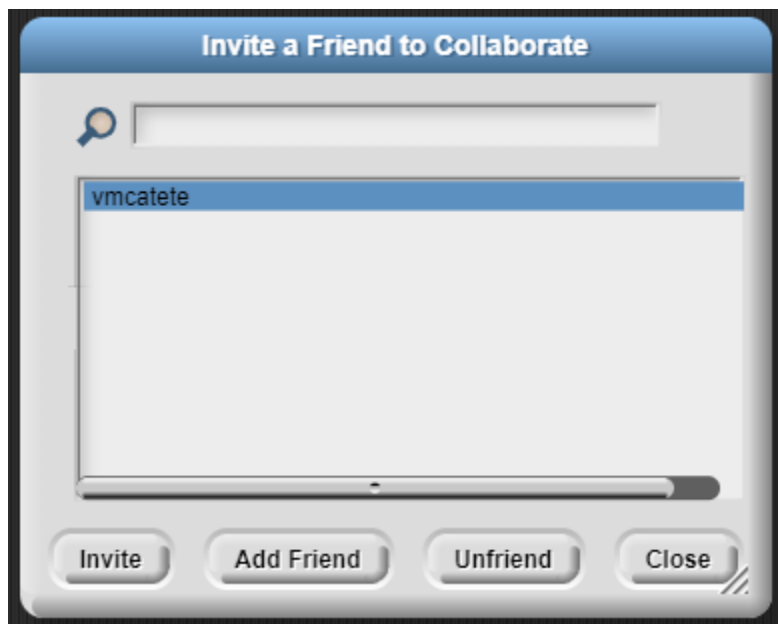
1. Use this url to access NetsBlox: <https://go.ncsu.edu/puzzlemethod>
2. Login to NetsBlox or create an account
3. Select the login menu in the top left of the screen [h]



4. Select "Friends" then "Send a Friend Request"
5. Enter your partners username into the text box and select "OK"
6. Select the login menu again and select "Collaborators"



7. Select your partners user name and select "Invite" to invite your partner to collaborate on the same file as you



8. Select the Puzzle piece icon towards the top right of your screen



9. Select "Hide Categories" and "Select User Categories"
10. Enter your or partners username into the text box, "Enter the username to configure".



11. Enter the categories you want to access into the next text box, "Available categories for <username>".

