ABSTRACT

Title of Dissertation:	COMPUTATIONAL THINKING IN EARLY GRADE CLASSROOMS: HOW YOUNG LEARNERS INTERACT WITH PHYSICAL DEVICES TO GROUND THEIR UNDERSTANDING OF COMPUTATIONAL THINKING
	Bih Janet Shufor Epse Fofang, Doctor of Philosophy, 2024
Dissertation directed by:	Associate Professor, David Weintrop, Teaching and Learning, Policy, and Leadership

Computational thinking (CT) has been supported as an important skill every young person should possess for the 21st century, with possible implications for problem-solving, self-expression, and creativity. Numerous initiatives, both within and outside classroom settings, have been developed in response to policy mandates aiming at broadening participation for all K-12 learners to acquire CT skills. Consequently, there has been a proliferation of computational toys and tools specifically designed for young learners, including codable robots introduced into classrooms and educational environments. With the growing prevalence of computational devices in educational settings, educators, curriculum designers, and researchers must cultivate diverse teaching approaches and deepen their understanding of how young learners engage with these devices to acquire CT skills effectively within classroom contexts. In this dissertation, I present findings of how elementary-grade learners develop CT skills when they program Sphero robots in mathematics classroom activities. I specifically focused on the kinds of representations

students developed, considering their *perspectives (understanding)* of the environment, and the *practices* they engaged in to accomplish given tasks.

To understand how young learners acquired CT skills, I observed fourth-grade learners as they interacted with activities on the Sphero.Math curriculum to program the Sphero robot in mathematics classrooms. The Sphero.Math curriculum was developed through a collaborative effort between researchers and DCPS partners. Findings from this work revealed that representations play an important role in supporting young learners to engage in CT practices such as Pattern recognition, algorithm design, problem decomposition, and abstraction (PRADA). Findings showed that representations such as (1) concrete manipulatives, (2) language, (3) graphic, (4) symbolic and (5) embodied representations provide scaffolds for learners to gain (PRADA), CT skills through iterating, testing, debugging, abstracting, modularizing, and reusing code.

Additionally, the design features of the Sphero robot and its programming environment support CT knowledge acquisition. Features such as (1) programmable LEDs provided opportunities for learners to break down tasks and create opportunities to organize and structure components to get visual feedback that helped them recognize patterns. (2) Taillight ("aim") LED provided visual cues, that facilitated the involvement of geocentric orientation and embodied practices that empowered students to establish sensorimotor references. (3) Sphero's virtual protractor supported students through the CT component of abstraction to address the geocentric aspects of the Sphero robot. (4) block-based environment/language, that involves the use of shapes and colors as effective visual aids and abstraction tools, to support the learners' construct to algorithms.

This research can serve as a resource for researchers, curriculum designers, educators, and designers to answer questions about design, choice of computational tools, and their respective programming environments that can afford meaningful CT experiences. Familiarizing learners with representations within CT robotics learning environments serves as a gentle initiation into emerging topics in education such as AI, ML, and data science, given the pivotal role representations play within these fields.

COMPUTATIONAL THINKING IN EARLY GRADE CLASSROOMS: HOW YOUNG LEARNERS INTERACT WITH PHYSICAL DEVICES TO GROUND THEIR UNDERSTANDING OF COMPUTATIONAL THINKING

by

Bih Epse Fofang Janet Shufor

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park, in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2024

Advisory Committee:
Professor David Weintrop, Chair
Dr. Tamara Clegg, Dean's Representative
Prof. Andrew Elby
Dr. Janet Walkoe
Dr. Caro Williams-Pierce

© Copyright by
Bih espse Fofang Janet Shufor
2024

Dedication

To my esteemed teachers, whose guidance and wisdom have illuminated my path throughout this academic journey. Your dedication to education and unwavering support have been invaluable in shaping my understanding and fueling my passion for learning. To my family, whose endless encouragement, patience, and understanding have sustained me through this endeavor. To my colleagues and friends, whose camaraderie, support, and encouragement have enriched my journey and made every challenge surmountable.

Acknowledgements

The completion of my Ph.D. journey at the University of Maryland, College of Education, in the Department of Technology Learning and Leadership, marks not only an academic achievement but a journey enriched with support, guidance, and inspiration from a multitude of remarkable persons. This acknowledgment is only but a humble attempt to express my deepest gratitude to all those who have been part of this incredible journey.

Thank you to my parents, who have always believed in me. Your unconditional encouragement in every endeavor I undertook has been the cornerstone of my success. Your absence at a time like this is heartbreaking. May you both continue to rest in peace knowing I am ok. You instilled in me the values of hard work, continuous learning, seeking happiness, and the importance of self-awareness as a precursor to understanding others. To my dear departed mother and father, Tawah Felicia and Nsoh Daniel; thank you for your guidance and, for supporting me in my growth as a person, an entrepreneur, and an educator. To my sons Nsoh, Mualah, Tsi, Tassah, and Miyaka, thank you for your encouragement and support. Thank you for standing by me even in the most difficult times in our lives, we've always found a way to stay on top. To my brothers and sisters, Rita, Julius, Jude, Mike, Emma, Divine, Gerald, and Quinta, thank you for being there for me and cheering me on through this journey. Your encouragement kept me going.

To my professors and staff at the University of Maryland, who have left an indelible mark on my academic and personal development. Your dedication, unique approaches to teaching, and willingness to share your knowledge have greatly enriched

my academic journey. I am most grateful to my advisor David, for allowing me to join his research team, and for diligently guiding and supporting me through my research endeavors. I would not be where I am today without his thorough and unwavering mentorship. Many thanks to Janet, Margaret, Peter, and all the DCPS collaborators who worked with us on the Sphero.Math project. This dissertation would not have been possible without your support and input to the project. I am also grateful to Caro and the Math play team, Nihal, Etka, Amber, and the YXLab of the iSchool team who have welcomed me into their labs and created a nurturing supportive environment where I could share ideas and learn.

To the TLL family, past and present, Virginia, LC, Merijke, Heather, Noelle, Alex, Jimmy, and Rotem, your support has not only made the journey lighter and enjoyable, but you helped hone my skills as a researcher through your feedback and shared experiences.

To my students, parents' colleagues, and entrepreneurial friends in Cameroon, this is for you! We have come a long way in our quest for excellence. This work is the fruit of our shared passion to seek for knowledge and innovation to develop our community, always pushing the boundaries of what is possible.

Charles, your openness to support me and the children, and extending your friendship beyond work and business has been a gift of immense value, and I am grateful.

This dissertation marks a significant milestone in my life and that of many women in my community. Ghanaian scholar Dr. James Emmanuel Kwegyir-Aggrey (1875-1927) said "If you educate a woman, you educate a family (nation)". This Ph.D.

means responsibility and at the same time opportunity for women and girls in my community.

Table of Contents

Dedication ii
Acknowledgements iii
Table of Contents vi
List of Tables ix
List of Figures x
List of Abbreviations xiii
Chapter 1: Introduction 1
Researcher Narrative 7
Contribution of this Dissertation 9
Structure of this Dissertation 10
Chapter 2: Literature Review 12
Introduction 12
Computational thinking and problem-solving 12
Computational Thinking for Young Learners 14
Educational Robotics 17
Programming Robots in Early-Grade Classrooms 19
Programming Robots and Computational Thinking 21
Computational Thinking and Representations 22
Representations 22
Computational Thinking 26
The Lesh Transition Model 27
Multiple Representations 29
Representations Manipulatives, and Symbols 32
Models and Modeling 35
Chapter 3: Methodology 38
Introduction 38
The Sphero.Math Project 38
Mutual Supportiveness of CT and Mathematics 41
The Sphero Robot 41
Study Design 46
Data Collection and Analysis 47
Settings and participants 47
Data Collection 49
Data Analysis Strategies 51
Analytic Approach 54
Chapter 4: Features of the Sphero Robot and its Accompanying Programming Environment that Facilitate Students' Understanding of CT Knowledge 56
Introduction of Sphero Robot as a CT tool used for Sphero.Math Activities 56
Overview of Sphero Robot 57
Features of the Sphero Robot and Vignettes Supporting CT 58
Sphero Programming LEDs 58
Vignette for Sphero Programming LEDs 61
Vignette Analysis 63

Sphero Taillight ("aim") LED 65
Vignettes for Sphero Taillight ("aim") LED 67
Vignette Analysis 69
Sphero Virtual Protractor 71
Vignette Analysis on Sphero Virtual Protractor 75
Sphero Robot Programming environment 79
Overview of Programming in Sphero Edu App 79
Supported programming languages for Sphero Robot 79
Sphero Block-Based Environment/Language 86
Vignette for Sphero Block-Based Environment 86
Vignette Analysis on Sphero Block-Based Environment/Language 90
Conclusion 96
Chapter 5: Findings 101
Use of Concrete Representations – Manipulatives and Objects 107
Description of Concrete Representations 107
Concreteness of Sphero.Math Materials and Description of the Activity 108
Vignettes with Students Using Concrete Representations 110
Interconnected Representations in Concrete Representations Vignette 119
Discussion 120
Graphic Representations 121
Description of Graphic Representations 121
Graphic Representations Activity 123
Language Representations 132
Description of Language Representations 132
Vignette Foregrounding Language Representations 135
Vignette Analysis for Language Representations 140
Discussion 143
Symbolic Representations 144
Description of Symbolic Representations 144
Vignette Script for Symbolic Representations 147
Explanation of Vignette 152
Symbolic Representations and Computational Thinking 156
Proficiency in the use of Symbolic Representations and CT can Lead to Creativity 158
Discussion 159
Embodied Representations 160
Description of Embodied Representation 160
Description of Activity Using Embodied Representation 162
Vignette Foregrounding Embodied Representation 164
Interconnectedness of Embodied Representations and other forms of Representations 173
Discussions on Embodied Representations 175
Interconnectedness of Representations 176
Description of Carousel Repair Activity 177
Analysis of Interconnectedness Vignette 180
Discussion 184

Conclusion 187
Chapter 6: Discussion and Conclusion 190
How Young Learners Interact with the Sphero Robot to Ground their Understanding of Computational Thinking 193
Interconnectedness of Representations in CT Environments 209
Variability in Representational Strategies Among Learners 211
Implications 212
Implications of these Findings for Teachers 212
Implications of these Findings for Designers 214
Implication of these Findings for Researchers 217
Considerations for Implications: Insights for Education Across Grade Levels 218
How these Findings Relate to Emerging Topics like AI, ML, and Data Science 219
Generalizability of programming Sphero robots in early-grade classrooms 221
Potential Avenues for Future Work 223
Limitations 225
Conclusion 226
Appendix A 231
Appendix B 293
Interview Questions 293
Appendix C 295
Analysis Codebook 295
Bibliography 324

List of Tables

Table 1. Educational Technology for CT Development 15
Table 2 Computational Concepts by Brennan & Resnick 16
Table 3. Three dimensions of computational thinking: computational concepts, computational practices, and computational perspectives.. 27
Table 4. Sphero.Math project CT component definition and examples 55
Table 5. Features of Sphero Blocks. 80
Table 6. Five forms of representations and their contextual uses 104
Table 6. Features of Sphero Blocks 107

List of Figures

Figure 1The Lesh Transition Model 29
Figure 2 Display data showing the distance Sphero traveled. 43
Figure 3 The Sphero robot (a) iPad and programming interface (b) Sample program (c) 45
Figure 4 Sphero roll block. 46
Figure 5 A Sphero Block program showing direction, speed, and time. 47
Figure 6 Sphero SPRK and its accessories. 58
Figure 7 Sphero LEDs on the robot and tactile interface panel on the Sphero app to select colors. 59
Figure 8 Sphero programmable LED block. 60
Figure 9 Monorail stops activity grid with instructions on “flashing lights”.. 61
Figure 10 (a) Aim (taillight) on Sphero robot (b) Heading set at 0 degrees on the Sphero interface. 66
Figure 11 Tactile panel for “aiming” the Sphero taillight. 67
Figure 12 Students use parts of their bodies to align and find orientation for the “aim” light on the Sphero. 69
Figure 13 Student walking on the triangle with the iPad and visualizing the virtual protractor. 72
Figure 14 (a) Sensor data icon. (b) Sensor data students queried to see the direction their robot travelled. 73
Figure 15 Student using the virtual protractor and physical protractor to support their understanding of angles on a triangle. 73
Figure 16 Arrow symbols within the virtual protractor representing various angles and directions indicating the robot's intended movement. 74
Figure 17 Students use tactile movements to rotate the arrow to select angles. 75
Figure 18 Students connect hand gestures to embodied practices to find the robot’s directions. 76
Figure 19 Students connect hand gestures to embodied practices to find the robot’s directions. 77
Figure 20 Students engage in embodied practice to understand the robot’s directions. 78
Figure 21 Sphero movement block, the “roll block”. 84
Figure 22 (a) Block programming interface showing the roll block with the possibility to view text code and sensor data. 85
Figure 23 The program provided an activity for students to use to determine the heights of their mates. 88
Figure 24 Students use colors and shapes to find and nest the distance block into a build-string block.. 92
Figure 25 Students swipe on the gray blocks on the sensors categories at the bottom of the screen to find the pill-shaped grey distance block. 93
Figure 26 Students opened the build-string blocks to edit their code using shapes and color codes to find the appropriate blocks to nest into other blocks. 95

Figure 27 Sphero.Math monorail activity sheet. 111
Figure 28 Monorail activity with a robot and iPad provided by the teacher at the start of the activity. 111
Figure 29 Students drag and drop codes onto the coding canvas. (b) Codes nested into the loop block. 114
Figure 30 (a) Students engage in the grid map as a concrete representation.. 115
Figure 31 Single lines of code written by students to navigate the monorail path. (b), (c) Students after recognizing a pattern used loops to nest blocks. 116
Figure 32 Grid map for monorail stops indicating colored squares and stops that serve as symbolic representations. 123
Figure 33 Students working on a maze with symbolic representations of Start and Stop indicated on the map. 125
Figure 34 Students write down actions the robot will have to make in a notebook. 130
Figure 35 The teacher stretches her arms to mimic the movements of a train. 136
Figure 36 The teacher shows the image of a monorail to students. 136
Figure 37 The teacher holds up the map of the monorail and projects the image of the map on the board. 137
Figure 38 Students translate language representations through writing to communicate the movements of the robot on the monorail path.. 140
Figure 39 Five full function codes to represent each of the five activities in the amusement park.. 148
Figure 40 Dot and Hashtag symbols were designed for the activity. 148
Figure 41 Color-coded instruction manuals designed by students to help visitors easily navigate the amusement park. 150
Figure 42 Five hashtags and dots represent the five activities designed for the amusement park.. 151
Figure 43 Amusement Park activity is modeled to represent amusement activities for visitors who visit an amusement park.. 152
Figure 44 The students use a blue taillight from the Sphero robot to represent the robot's heading or direction, which they call "aim." 154
Figure 45 Sphero.Math activity model for Triangles. 163
Figure 46 Students use their bodies to understand abstract concepts like angles. 164
Figure 47 A student uses both his finger and hand to show the intended direction he would like to program the Sphero to move. 165
Figure 48 Student supports his understanding using a virtual and physical protractor. 165
Figure 49 Student uses gestures to communicate with his teammate the direction the robot should move. 166
Figure 50 Student maps the triangle first using his right hand. 168
Figure 51 The student uses her hand and finger to gesture and embody both the robot and the angle. 169
Figure 52 Students move their fingers on the inner edges of the triangle mapping out the angles. 174
Figure 53 Carousel repair activity on Sphero.Math curriculum. 178
Figure 54 Carousel repair activity sheet. 179

Figure 55 Interconnected meaningful connections of representations from verbal to symbolic and embodied representations a student navigates to problem solve..... 181
Figure 56 Student embodies the code and a physical representation of the robot.... 183
Figure 57 Function block, define block with its child codes that carry the instructions the function will execute..... 184
.

List of Abbreviations

AI Artificial Intelligence

CS Computer Science

CT Computational Thinking

DCPS District of Columbia Public Schools

LED Light Emitting Diodes

ML Machine Learning

PD Professional Development

RPP Research Practitioner Partnership

UMD University of Maryland – College Park

Chapter 1: Introduction

Computational thinking (CT) has been held up as an important skill set for young people to possess for the 21st century, with implications for problem-solving, self-expression, and creativity (Wing, 2006). Several initiatives have been designed to support children develop CT skills, some through interactions with codable robots. The extent to which computational devices and tools permeate our society may be a call to developing varied approaches to teaching CT skills to everyone, not just for computer scientists or those working with technologies related to computing. As Jeannette Wing would put it, "*CT is a fundamental skill for everyone, not just for computer scientists. To reading, writing, and arithmetic we would add CT to every child's analytical ability"* (Wing, 2006). To this effect, CT has been integrated into K-12 education in many NGSS lead states and has been posed as an important learning objective for all K-12 learners (NGSS Lead States, 2013). There have also been efforts to bring CT into classrooms by researchers either through calls to broaden participation or through providing opportunities by developing CT curricula where students can learn and develop basic CT skills (National Research Council, 2010; Shute et al., 2017; Wing, 2006; Weintrop et al., 2021). Most of the research around integrating CT in classrooms has been either through plugged or unplugged activities, integrated into other disciplines or as a stand-alone subject (e.g., Clements, 1999; Barr & Stephenson, 2011; Sengupta et al., 2013; Weintrop et al., 2016; Wilensky et al., 2014, Bih et al., 2020).

In essence, many of the authors cited above argue that these studies aim at understanding how students learn in CT enhanced environments and how they can build their own understanding of CT elements using the resources that are put at their disposal. CT enhanced activities may vary depending on the tools and environments designed to enact these activities. Some activities could be virtual, with activities presented in virtual formats, while others can be a combination of virtual and physical activities such as programming physical devices on computers, iPads, or phones. In both cases, certain elements of CT have been suggested as comprising CT and have been used to build the basis for learning and assessment in CT-enhanced environments. For example, the PRADA (pattern recognition, abstraction, decomposition, and algorithms) model for integrated CT (Dong et al., 2019), and other CT practices that support programming activities, such as iterative development and debugging (Weintrop et al., 2022).

While these elements can be noticed in CT-enhanced activities, a critical look at the activities and research around educational computing suggests that CT activities in k-12 classrooms have mostly involved the use of technologies. This calls for considering learners' experiences with technology as building blocks for a deeper understanding of the kind of learning and the development of disciplinary expertise in either CT or other disciplines the learners have made (e.g., Papert 1980; DiSessa 2000; Sengupta, 2018). Learning objectives in these technology-led CT environments have predominantly focused on the development of programming skills, STEM skills, or the production and use of other computationally related skills (e.g., see the studies reviewed by Grover and Pea (2013a). One reason could be that CT takes its origins

from computer science and can be applied across disciplines to support problem-solving contexts and activities. In addition to its "support abilities" in learning and problem-solving contexts, CT has been argued to be an "analytical ability essential to every child" (Resnick et al., 2009, p.62). CT has also been suggested as critical to raising academic achievement and solving real-life problems for K-12 students (Sykora, 2014). However, efforts to develop tools and activities to support the development of CT in young learners are gaining steam (Saez-Lopez, Roman-Gonzaez, & Vazquez-Cano, 2016). Readily available educational technologies to develop CT skills have previously focused on computers and computing devices, requiring knowledge and command of programming languages typically taught in middle and high schools (Ching et al., 2018). Educational tools to teach CT to young learners include toys, robots, and board games. Young children have access to computational devices both at home and at school, making the introduction of CT into early grades possible.

The research community has in several studies, used educational robotics to teach CT skills to young students (Papert, 1980; Resnick, 2012; Stoeckelmayr et al., 2011; Bers, 2010; Bers et al, 2014; Kazakoff, Sullivan, & Bers, 2013; Alimisis & Kynigos, 2009; Benetti, 2012; Bredenfeld, Hofmann, & Steinbauer, 2010; Johnson, 2003; Boticki, Pivalica, & Seow, 2018). Authors in these studies use different programmable robots to investigate how young children interact with these robots and their designed environments to develop CT skills. Through programming, children engage with tools and artifacts to "write" new types of things and do not "just learn to code, they code to learn" (Resnick et al., 2009; Resnick, 2013, Para.5). Put in context,

several developmentally appropriate programming languages have made it possible for children to program tangible artifacts, making it possible for CT to be introduced to young children, and even integrated into classroom activities in the form of manipulatives and tools (Bers, 2008; Bers, 2018a; Bers & Horn, 2009).

Programming robots and other tangible artifacts allows children to be introduced to programming concepts such as debugging, sequencing, patterns, logic, loops and sometimes sensors, actuators, and motors. Other than programming and CS skills, researchers have also argued that shaping technological materials that are age appropriate can support CT for early learners and support creativity and inquiry-based learning for young learners while supporting learning (Bers, 2018a; Bers, 2012; Kazakoff & Bers, 2012; Sullivan & Bers, 2015; Bers, 2018a; Horn Crouser & Bers, 2012). These interactions with technological tools are important aspects to consider in the CT literature as they can either contribute to the development or constitute some elements of CT in these contexts.

From an epistemic perspective, CT practices have also been argued by Sengupta, (2018) to be centered around "knowing" and "doing" and heavily grounded around representations. Representations in these CT environments are embedded in the *concepts* and *designs* learners can engage with as they program their robots. In this context, programming robots can be a good environment to observe how young learners can express ways of knowing and doing. Tangible computing draws on the benefits of making "coding languages" exist in physical life in learning spaces. When robots are programmed, their codes automate activities for the robots to perform in visible formats and can provide real-time tinkering, debugging, and refinement in both the virtual and

physical space. Children can make sense of what is happening in real time, understand it, and fix it thereby making learning possible at the same time. The added visibility of code in these environments can be made possible for children by how accessible the tools are to young children.

The kinds of representations students develop and navigate are usually based on their *perspectives (understanding)* of the environment around them, and the *practices* they engage to accomplish given tasks. According to Brennan and Resnick (2012), children may engage in specific programming concepts such as debugging, repeating, loops, or sequencing. Practices may include and are not limited to problem-solving, collaboration, and communication. In CT environments it can be challenging to understand children's perceptions of how things work. Perception may include questioning how the interface works to control the robot or how a robot change direction? It is in the process of observing how such analytical and critical thinking develops into problem-solving skills in CT environments that I think we can understand how young children without experience in programming build knowledge through their own understanding of how to navigate CT based tasks.

To further explore how young children develop CT skills in robotics programming environments, in this dissertation I pursue the following questions:

- *What features of the Sphero robot and its accompanying programming environments facilitate students' understanding of CT knowledge?*
- *What kinds of representations and expressions do young learners engage in order to interact with Sphero robots in a mathematics classroom and how do these representations contribute to successful problem-solving in CT tasks?*

The first research question is design-centeric, looking at the features of the Sphero robot, the Sphero programming environment, and the design of the activities that support learning. This study considers the implementation of a CT curriculum aimed at supporting the development of CT skills within a classroom context. In curricula contexts, scaffolds are required to meet the expectations of every learner in the classroom. The design of the activities and choice of robot is therefore very important to consider in these circumstances. In addition to the knowledge and skills developed in this environment, it is important to understand the role of the design of the materials, smart artifacts and robotics and interface designs that support the development and integration of technological thinking among young children. In my second research question, I analyze how design aspects intentionally designed into the Sphero robot and its programming environment contributed to informing the kinds of representations learners engaged in the learning environment.

My second research question addresses aspects of student learning, representations, and interactions with the Sphero robot during Sphero.Math activities. This question addresses aspects of knowledge and skills involved in navigating the activities and programming the robots. In addressing aspects of knowledge and skill acquisition in question two, I define "successful problem solving " as instances when learners successfully program the robot to navigate given tasks on Sphero.Math curriculum lessons. I also consider instances when learners expressed knowledge about navigating challenges on complex programming and mathematical concepts and how they overcame some of the obstacles as "successful problem-solving" instances. I investigate representations that support the development of CT in young learners as

they program the Sphero robot in mathematics classes. These aspects include but are not limited to knowledge and skills in areas such as programming, perception of the robot's behavior, mathematics concepts, behaviors of the robot, and the representational structures they generate to control the robot.

Young children may lack programming skills and language to program, control, and automate a robot to perform tasks, they may resort to developing perspectives that can fit into representational structures to support their understanding of respective tasks. Some of these representations may be spontaneous, produced in a one-time episodic manner or become reusable over time; for example, children are seen over time representing a straight line or zero degrees by stretching their hand(s) straight in front of them to represent zero degrees or meaning the robot is going to roll straight in front of them. My second research question, therefore, should be able to help provide evidence for the ability upper elementary grade (4^{th} grade) learners can develop to navigate abstract concepts in both mathematics and CT and use representations to break down abstract concepts in mathematics and CT such as rule of behaviors and geometry. This study will contribute to literature that supports the idea that the design of age-appropriate features for robots and accompanying programming environments can lower the floor for young learners to engage in different forms of CT expressions and representations.

Researcher Narrative

Acknowledging the influence of my identity and personal background on my research endeavors, I offer the following statement regarding my positionality as a researcher. I am currently pursuing my Ph.D. at the University of Maryland, College

Park (UMD). For the past four years, I have worked on the Sphero.Math project. Within this project, I have helped with the development and implementation of the Sphero.Math curriculum in fourth-grade mathematics classrooms within DC Public Schools (DCPS).

Before coming to the University of Maryland, I served as an electronics teacher, in Cameroon where I explored opportunities for integrating electronics and interactive technology tools to enhance learning in low-resource classrooms. Embracing a philosophy rooted in constructionism, I inherently prioritize hands-on, experiential learning experiences for students using tools like robots to enrich their learning experiences. I recognize that my personal affinity toward constructionism serves as a lens through which I approach classroom instruction, inevitably influencing certain outcomes of my work.

Drawing insights from my teaching experiences, my research endeavors are deeply informed by practical insights gained from the classroom. I believe the findings from my research will inform and enrich my teaching practices moving forward. Through my involvement in the Sphero.Math project, I have gained valuable insights into how students learn during interactions with computational devices, the significance of embedding real-world scenarios into educational curricula, as well as the transformative power of collaborative group work.

Observing students engage in reflective practices and knowledge sharing during the Sphero.Math lessons have reinforced my belief in building teams with shared understanding within the classroom. I firmly believe that meaningful learning thrives within an environment characterized by shared interests and collaborative endeavors. I

remain committed to prioritizing teamwork and collaboration among student teams within my classroom, providing a safe and inclusive space for learners to exchange diverse perspectives, learn from one another, and collectively think about innovative solutions to real-world challenges within their communities.

In essence, my research endeavors are driven by a commitment to empowering learners to leverage their acquired knowledge in meaningful ways to address everyday problems within their communities, thus fostering a culture of impactful, community-driven innovation.

Contribution of this Dissertation

This study mainly contributes to our understanding of how representations can be used as scaffolds to support young learners in learning to "think computationally." This study has the potential to open research opportunities within the realm of representations in (CT) environments. By shedding light on patterns, relationships, and underlying structures supporting various forms of representations in CT environments, this study offers valuable insights. These insights can inform the design and implementation of effective strategies that support conceptual understanding of representations among educators, researchers, and learners, thereby contributing to the advancement of knowledge in this field. Teachers, educators, researchers, and curriculum developers, tasked with designing CT activities for young learners can benefit from this study, through understanding what CT practice looks like in robotics environments and how knowledge structures are built within these environments.

External representations have been extensively studied in mathematics literature and have opened possibilities to talk about mathematical relations and

meaning without always using inferences that concern individual students or the person solving the problem's experiences. Likewise, this study could also be an entry point for CT and the learning community to start considering a framework for describing individual knowledge structures and problem-solving processes in CT environments as a starting point for making generalized claims.

This work will also have design implications both for the educational technology industry and computer education research, entities with a common goal of facilitating student learning of computational knowledge and practices.

Structure of this Dissertation

This dissertation is organized into six chapters. The following sections (chapters 2 through 6) are organized as follows: After introducing the work in the first chapter, the second chapter proceeds with a literature review focused on CT and representations, with a specific emphasis on CT within the context of robotics environments and young learners. In chapter three, I describe the methods I use to collect and analyze the data collected for this study. I present the study design, data collection strategies, and the school setting from which the data is being collected.

Chapter four answers research question 1 that seeks to understand; *What features of the Sphero robot and its accompanying programming environments facilitate students' understanding of CT knowledge?* I explore how the design of the tools (robot and its programming environment) supported learners navigate CT tasks. Chapter five focuses on findings to answer my second research question that seeks to understand *What kinds of representations and expressions young learners engage to interact with Sphero robots in a mathematics classroom and how these representations*

contribute to successful problem-solving in CT tasks. I provide evidence on the kinds of representations learners engaged with and how those representations supported CT practice. Chapter five answers research question 2 that seeks to understand; *What features of the Sphero robot and its accompanying programming environments facilitate students' understanding of CT knowledge?* Chapter six offers a summary of chapters 1 through 5. I outline my research contributions and suggest potential future directions.

Chapter 2: Literature Review

Introduction

In this chapter, I review the literature on CT, programming robots in early-grade classrooms, and representations to explore how these three elements combined into classroom activities may contribute to answering the research questions. First, I will describe how to integrate technological tools into classrooms to promote experiential learning through hands-on practical activities using the constructionist approach to learning (Papert, 1980). Using robots to teach STEM or CT concepts was pioneered by Seymour Papert when he introduced robots and computers to engage with mathematical ideas and concepts. This process encouraged students to discover, build, and create new knowledge and understanding through practical activities. Since then, several studies have been carried out to analyze the pedagogical implications and potential robots can bring into K-12 classrooms (e.g., Lee et al., 2011; Bers & Horn, 2010; Mioduser & Levy, 2010; Resnick, 2006; Resnick et al., 2009; Haugland, 1992; Clements & Sarama, 2002).

Computational thinking and problem-solving

In this section, I present some literature about how CT relates to problem-solving abilities. As mentioned earlier, the idea of CT has been expanded from Papert's work since the early 1980s when he argued that programming computers could develop children's cognitive abilities. Papert further argued that programming computers allowed children to develop problem-solving skills and at the same time represent their

ideas in virtual environments. Wing, (2006) expanded CT as “a systematic way of thinking that can lead to the development of problem-solving skills, designing systems and understanding human behavior, by drawing on concepts fundamental to computer science” (P.33).

To further investigate the idea of problem solving in CT environments, Shute et al, (2017) systematically reviewed 70 documents and proposed a theoretical framework for CT based on problem-solving. Shute et al in this work, defined CT as “the conceptual foundation required to solve problems effectively and efficiently (i.e., algorithmically with or without the use of computers) with solutions that are reusable in different contexts”. Shute et al, (2017) further suggested that CT definitions may sometimes vary in how they are operationalized and are not particularly generalizable (e.g., Berland & Wilensky, 2015; Ioannidou, Bennett, Repenning, Koh, & Basawapatna, 2011; Israel et al., 2015). Other scholars have also defined CT in educational settings where physical manipulative tools are involved as a systemic way of thinking that provides opportunities for problem-solving skills to develop as well as a language for expressing ideas to emerge (Bers, 2018; Byers and Walker, 1995).

In addition, our research team provided a working definition for CT within robotics programming and mathematics activities as “the skills and practices associated with formulating a problem and expressing its solutions in such a way that takes advantage of computers and the things they are good at”.

These definitions point to the idea that problem solving is an important term to pay attention to in the CT literature. This study will rely heavily on the definition of Shute et al, (2017). This involves among other things, “learning about " and using

representations in the form of language and symbols they construct to communicate effectively to problem solve (Bers, 2018). The goal of this study is to understand the nature of these representations and expressions and how they contribute to successful problem-solving in CT tasks.

Computational Thinking for Young Learners

Computational thinking over the past years has been gaining traction in K-12 settings. Several efforts have been made to introduce and develop CT in young learners in the K-12 settings (Saez- Lopez, Roman-Gonzaez, & Vazquez-Cano, 2016; Barth-Cohen, Jiang, Shen, Chen, & Eltoukhy, 2018; Chen et al., 2017; Angeli, Xerou, & Nicolaou, 2019; Constantinou & Ioannou, 2018; Caballero-Gonzalez, Mũoz-Repiso, & García-Holgado, 2019). Several avenues have been explored to develop CT in young learners using educational technologies available for these age groups. Some of these technologies include traditional computers and computing devices with games and age-appropriate activities, toys, programmable robots, and board games. Some of these technologies have been reported to support learners learn computational concepts such as sequencing, loops, iterations(repetitions) parameters, conditional statements, branch, functions etc. (González & Muñoz-Repiso, 2018; Strawhacker & Bers, 2015; Sullivan & Bers, 2013).

To further support the introduction of computational tools for young learners to support CT development, Ching et al, (2018) developed a framework for analyzing tools in educational technologies for young learners that support CT development. This framework addressed two dimensions; (1) design of the tools and (2) the pedagogical affordances of these tools. Educational Technology CT tools could be designed to

either live only on the screens as screen-based programming tools or designed as tangibles (concrete manipulatives).

Kafai and Burke, (2014) explored the potential of educational technologies to support programming for young children and concluded that these tools have moved "beyond the computer screen to meld the digital with the tangible" (p. 91). In terms of design, a growing number of educational technology products have been designed to develop CT and programming skills in young learners. These products are classified by Ching et al, (2018) in the table below. According to Ching et al, (2018), programmable manipulatives are designed to target very young learners compared to non-manipulatives for programming that are designed to target learners with a stronger command of reading and writing, mostly students in higher grades.

Table 1. Educational Technology for CT Development (source: (Ching et al., 2018))

	Programming with manipulatives	No manipulatives for programming (Programming on screen)
Electronic Physical Enacting Agents	**Programming Toys** • Think & Learn Code-A-Pillar • Cubetto • Code & Go Robot Mouse Activity Set • KIBO	**Robot Kits** • Dash & Dot • LEGO® WeDo 2.0 • LEGO® Mindstorm
No Electronic Physical Enacting Agents	**Board Games** • Robot Turtles • Code Monkey Island **Augmented Reality Tools** • Osmo Coding	**Programming Concept Practicing Applications/Websites** • Kodable • Lightbot • Lightbot Jr. • Cargo-Bot • Code.org **Animation/Game Development Tools** • Scratch • Scratch Jr. • Hopscotch • Tynker: Coding for Kids

Regarding pedagogic affordances to develop CT skills in education, Brennan and Resnick, (2012) developed a framework to understand CT skills in education. This framework defines three dimensions of CT (1) "computational concepts (the concepts

designers employ as they program), (2) computational practices (the practices designers develop as they program), and (3), computational perspectives (the perspectives designers form about the world around them and about themselves)" (Brennan & Resnick, 2012, p.3). While these concepts described in this framework will contribute to the CT concepts students will develop, their interactions with the tools and the setting of the lessons and classroom interactions will also play a role in the development and expression of CT concepts. Table 2 shows different computational concepts Brennan & Resnick (2012) highlight in their framework.

Table 2 Computational Concepts by Brennan & Resnick (2012)

Table 2. *Computational Concepts.*

Concept	Description
Sequence	Creating a series of individual steps or instructions that can be executed by the computer
Loops	Running the same sequence multiple times
Parallelism	Making things happen at the same time
Events	One thing causing another thing to happen
Conditionals	Making decisions based on conditions
Operators	Supporting mathematical and logical expressions
Data	Storing, retrieving, and updating values

Educational robots, on the other hand, have been reported to encompass a wide range of pedagogical affordances by combining screen-based features and tangibility. Robots as instructional education technology tools have been described as *enacting agents* (Ching et al., 2018). These enacting agents could take the form of screen-based programming tools or tangibles. Ching et al (2018), further argue that manipulatives for programming can enable learners to express their thought and programming ideas through tinkering, sorting, arranging, interacting, and exploring manipulatives. In the

section below, I review the literature on educational robotics, programming robots in early-grade classrooms, programming, CT, and problem-solving.

Educational Robotics

The idea of introducing computational technologies in classrooms stemmed from Papert's argument that constructionist programming environments are powerful tools to engage children to transform abstract ideas into concrete ideas (Papert, 1980, 1993). Papert argued constructionist programming environments can be powerful engaging environments for children to learn with objects (Papert, 1980). Papert created the LOGO programming language, and studies found that children in kindergarten who used LOGO work, were attentive, became more creative, and had a sense of self-direction (Clements & Meredith, 1992). Programming has also been reported to support the development of cognitive and problem-solving skills (Haugland, 1992; Clements & Sarama, 2002). Programming robots to perform tasks in classroom contexts have added to the list of innovative classroom activities to improve learning.

Even though young children may not possess the syntactic skills required for effectively writing programming languages, they can acquire programming knowledge by translating between the physical and virtual environments of the activity using various forms of representations (Burleson et al., 2018). Such interactions with technological tools provide opportunities for young learners to start developing an interest in programming, developing CT skills, and taking interest in STEM activities from an early age - practices that have been reported to foster problem-solving, learning, motivation, and creativity (Marina U Bers, 2005; Pea & Kurland, 1984; Resnick, 2012).

Programming physical devices and interfaces can play an important role in the outcome of children's programming abilities. These physical devices are programmable manipulatives that can take the form of plugged robotics toolkits such as beebots, code and go mouse, code a pillar etc., unplugged CT toys and board games (see table 1). Children programming across multi modalities such as animations, graphical models, games, and robots using age-appropriate materials can learn and apply core CT concepts such as iterative design, decomposition abstraction, automation, analysis, modularization (e.g., Lee, et al., 2011; Bers & Horn, 2010; Mioduser & Levy, 2010; Resnick, 2006; Resnick et al., 2009). As part of integrating constructionist and programming activities into early grade classrooms, a good number of studies have cautioned about the importance of scaffolding these activities using developmentally appropriate tools and interventions in order to meet the cognitive expectations of learners at different stages of their development (Marina U Bers, 2005; Marina Umaschi Bers, 2010; E. Kazakoff & Bers, 2012; Mioduser & Levy, 2010; Resnick, 2012; Resnick et al., 2009).

One way of understanding how programming physical robot activities can contribute to acquiring meaningful programming skills for young learners requires recognizing the existence of the types of programming environments different robots provide. This also means uncovering detailed investigations of how young children with limited technological abilities engage with robots in classroom and laboratory settings. Such investigations can provide a better understanding of the nature of programming environments educational robots provide their learning affordances for

young children, and the nature of interactions these educational robots currently support.

Programming Robots in Early-Grade Classrooms

The idea of introducing computational technologies in classrooms stemmed from Papert's argument that constructionist programming environments are powerful tools to engage children to transform abstract ideas into concrete ideas (Papert, 1980, 1993). Papert argued constructionist programming environments can be powerful engaging environments for children to learn with objects (Papert, 1980). Papert created the LOGO programming language, and studies found that children in kindergarten who used LOGO work, were attentive, became more creative, and had a sense of self-direction (Clements & Meredith, 1992). Programming has also been reported to support the development of cognitive and problem-solving skills (Haugland, 1992; Clements & Sarama, 2002). Programming robots to perform tasks in classroom contexts have added to the list of innovative classroom activities to improve learning.

Even though young children may not possess the syntactic skills required for effectively writing programming languages, they can acquire programming knowledge by translating between the physical and virtual environments of the activity using various forms of representations (Burleson et al., 2018). Such interactions with technological tools provide opportunities for young learners to start developing an interest in programming, developing CT skills, and taking interest in STEM activities from an early age - practices that have been reported to foster problem-solving, learning, motivation, and creativity (Marina U Bers, 2005; Pea & Kurland, 1984; Resnick, 2012).

Programming physical devices and interfaces can play an important role in the outcome of children's programming abilities. These physical devices are programmable manipulatives that can take the form of plugged robotics toolkits such as beebots, code and go mouse, code a pillar etc., unplugged CT toys and board games (see table 1). Children programming across multi modalities such as animations, graphical models, games, and robots using age-appropriate materials can learn and apply core CT concepts such as iterative design, decomposition abstraction, automation, analysis, modularization (e.g., Lee, et al., 2011; Bers & Horn, 2010; Mioduser & Levy, 2010; Resnick, 2006; Resnick et al., 2009). As part of integrating constructionist and programming activities into early grade classrooms, a good number of studies have cautioned about the importance of scaffolding these activities using developmentally appropriate tools and interventions in order to meet the cognitive expectations of learners at different stages of their development (Marina U Bers, 2005; Marina Umaschi Bers, 2010; E. Kazakoff & Bers, 2012; Mioduser & Levy, 2010; Resnick, 2012; Resnick et al., 2009).

One way of understanding how programming physical robot activities can contribute to acquiring meaningful programming skills for young learners requires recognizing the existence of the types of programming environments different robots provide. This also means uncovering detailed investigations of how young children with limited technological abilities engage with robots in classroom and laboratory settings. Such investigations can provide a better understanding of the nature of programming environments educational robots provide their learning affordances for

young children, and the nature of interactions these educational robots currently support.

Programming Robots and Computational Thinking

One of the reasons authors have advanced for bringing robots into classrooms is for the development of CT skills, especially in the early years. For example, several studies have used programming robots to develop CT skills for young learners (Angeli, Xerou, & Nicolaou, 2019; Constantinou & Ioannou, 2018; Caballero-Gonzalez, Mũoz-Repiso, & García-Holgado, 2019).

These studies positioned activities involving programming robots as an effective way for developing CT skills either through stand-alone robotics activities or interdisciplinary activities where programming robots are used to introduce concepts in other disciplines. In addition, some of these studies designed classroom interventions that either involved a robotics curriculum activity or employed a robotics curriculum to support an existing curriculum of targeted disciplines (Marina U Bers, 2005; Marina Umaschi Bers, 2010; E. Kazakoff & Bers, 2012).

In a bid to advance CT skills, some studies have focused on CS programming concepts like sequencing, debugging, repetitive commands, variables, conditional statement's function, and iterations as part of programming and CT skills (Grover & Pea, 2013; Weintrop et al., 2016). Studies investigating CT skills in CS contexts have either used interventions that focused on problem-solving by performing tasks designed to understand problem-solving strategies while programming a robot or developed and used robotics curricula designed to assess CT frameworks.

For example, Barth-Cohen and colleagues, (2018) proposed a framework to conceptualize how elementary students interpret and navigate multiple representations in CT problem-solving. Additionally, Chen et al. (2017) adopted a curriculum in their intervention that focuses on an assessment instrument using CT frameworks. Problem-solving interventions have mostly involved tasks designed on mats, e.g., mazes and challenges to explore CT skills during robotics programming are common activities used to explore CT concepts (Magana et al., 2016; Angeli, Xerou, & Nicolaou, 2019; Constantinou & Ioannou, 2018).

Computational Thinking and Representations

Given the centrality of representations to answer my second research question, I review the literature on representations broadly and then narrow it down to literature on CT.

Representations

According to Moore et al, (2020), the ability for students to problem-solve in a CT environment depends on their ability to navigate between multiple representations. This ability to translate across multiple representations is called representational fluency and is a highly important skill for students to possess to develop a deeper understanding of concepts (Greenes & Findell 1999; Johri & Olds 2011; Lesh & Zawojewski 2007; Suh & Moyer 2007). In broader terms, Goldin, (2008) describes representations as a setting that can represent something somehow. According to Goldin, the term "representation" encapsulates both the process of "representing" and the outcome of the process involved in the representation. Representations are seen as

privileged tools in mathematics education, because of the flexibility they offer students to express their mathematical ideas while helping them construct new knowledge in the process (Serrazina, 2017). Representations, however, cannot live on their own. They must always be a part of a structured system, interconnected with other representations to be able to make meaning (Goldin & Shteingold, 2001).

Representations, and the fluency to navigate through them, can provide cognitive support in computational environments and demystify abstract information that may otherwise seem inaccessible to young learners (Magana et al., 2016; Wing, 2006). More so, representations are not uniform across contexts. Depending on the activity, representations can take different forms and structures to fit the context and new representations can be constructed whenever necessary (Ainsworth, 2006). For example, in play activities, it is usually the child who invents the symbolic mapping to be able to relate to the meaning they intend the symbol to represent; During play, children may simulate car actions by moving a piece of wood back and forth (at that moment of play, the piece of wood represents a car). It is usually the child's idea that guides his/her choice of representations per object or model found in the activity (Deloache,1991).

Representations in play may differ from representations in designed activities (e.g., curriculum-structured activities with set rules and tasks to accomplish) (Deloache,1991). When a child in a playful activity manipulates an object, he already knows what the object will represent, meaning he/she is in a second iteration of what the object represents other than being what it is. In a modeled (none play) activity, it may be a little more complex as the child may have to generate representations to fit

the models or activities designed (Deloache, 1991). Modeled activities require children to discover the symbolic relationships created for the activity. For example, in a curricular activity that involves manipulating objects, the activity tries to convey to the child, the researcher, and the teacher's idea of how the model and the environment around the model should relate (Deloache, 1991; Ainsworth, 2006; Lesh and Zawojewski 2007). In other words, the learner's thinking is guided, making it harder for the learner as they must think through the model in two ways; first as a symbol, then as an object, with both representations being active at the same time. In CT robot programming environments, it is therefore important to realize the level of complexity in terms of cognitive engagement and effort that learners need to understand and interpret symbols and translate them between physical and virtual environments.

Research on gestures and symbol representation as simulated actions argue gestures and representations do not happen by coincidence. The Gesture as a Simulated Action GSA framework (Hostetter and Alibali, 2008), which holds that representational gestures are produced when actions are simulated as part of thinking and speaking, claims that motor simulations are involved in thinking about actions, thereby resulting in representational gesture. Likewise, in symbolic representation activities, there is first, the awareness of the existence of a symbolic representation, that what they are interacting with may mean something else and that relation is not immediately clear in the onset for younger children. Deloache (1991) posits that children need prior experience with certain symbolic representations they already understand to successfully appreciate and understand other symbolic representations leading to the conclusion that symbolic development may happen because of

cumulative experiences with symbols. Children can therefore construct meaningful knowledge after having achieved “representational insights" from one symbolic reaction – meaning that, the process of learning about new insights in a domain or concept may eventually emerge through interactions with representations.

Mature symbol users however can possess skills to make comparisons with other similar objects and adopt representations for a wide range of things in ways that novice symbol users may not necessarily possess (Burns, 1990; Deloache & Burns, 1989). This supports the idea that children in CT and programming environments may develop representations because of their lack of experience in understanding and use of programming languages. Their translation between these representations can help them develop representational fluency (Fofang et al., 2021; Moore et al., 2020).

Given the importance of understanding how students use representations in this study, I will use the Lesh Translation Model (LTM), a framework for understanding the use of representations and representational fluency to operationalize representations and translations (explained in detail below). I will also use the Gestures as Simulated Action (GSA) framework by Hostetter and Alibali, (2008) to understand representational gestures produced when actions are simulated as part of thinking and speaking. This framework will help me to address the use of language and gestures, pointing, depicting objects with the shape of hands, and indicating spatial locations as the representation of thought to convey ideas (Hostetter and Alibali 2008). Gestures and body movements also referred to as embodied cognition and actions, as described by mathematics education literature can support learning and recall, development of

language, and likewise the development of concrete and abstract concepts (Alibali and Nathan 2012; Macedonia and von Kriegstein 2012; Roth and Lawless 2002).

Computational Thinking

Several studies have recognized CT as vital for developing 21st century skills. This goes beyond using tools. It involves creating tools and understanding the implications of these creations (Magana et al., 2016). CT involves integrating disciplinary knowledge with cognitive processes to formulate and solve problems, presenting solutions in a format executable by information-processing agents, using concepts like abstraction, recursion, and iteration to process data and generate artifacts (National Research Council, 2010; Shute et al., 2017; Wing, 2006; Weintrop et al., 2021; Barr & Stephenson, 2011; Sengupta et al., 2013; Weintrop et al., 2016; Wilensky et al., 2014, Bih et al., 2020).

Additionally, it has been argued that CT is more than just problem solving, as it requires students to develop algorithmic thinking to be able to solve problems systematically. Such skills require a certain level of fluency and language to communicate ideas within the situative "in the moment" nature of language and interactions learners engage in as they work in robot programming environments (Papert, 1980; Bers, 2010). Identifying different categories of representations is important in terms of framing how students' representational systems work together and develop during problem-solving processes. Representations and symbol systems have been widely acknowledged as integral to the concept of CT and can serve as one of the foundations for curricula designed to facilitate learning and evaluate its progression (Grover & Pea, 2013; Weintrop et al., 2016). In this study, I analyze how

learners create and translate across different representations (concrete manipulatives, graphic representations, verbal representations, symbolic representations, and embodied representations).

Furthermore, for this study, I use aspects of CT dimensions that break down CT concepts into aspects such as sequencing, loops, conditionals, events, etc. (table 4). I also identify CT practices as tied to aspects such as iteration, testing debugging, reusing, remixing, abstracting, and modularizing. I analyze perspectives of CT use in ways the learners express and ask questions during the activities (Lye & Koh, 2014).

Table 3. Three dimensions of computational thinking: computational concepts, computational practices, and computational perspectives. (Source: S.Y. Lye, J.H.L. Koh /Computers in Human Behavior 41 (2014) 51–61.

Dimension	Description	Examples
Computational concepts	Concepts that programmer use	Variables Loops
Computational practices	Problem-solving practices that occurs in the process of programming	Being incremental and iterative Testing and debugging Reusing and remixing Abstracting and modularizing
Computational perspectives	Students' understandings of themselves, their relationships to others, and the technological world around them	Expressing and questioning about the technology world

The Lesh Transition Model

The Lesh Transition Model (LTM) (see figure 1) is a framework for understanding the use of representations and representational fluency. The LTM is built on the three different types of representations used by Burner, (1966) to understand the world: representations through doing, representations through depicting, and representations through symbols. Building off these three forms of representations, Lesh and Doerr, (2003) went further to make five forms of representations; representations through Concrete representations, realistic metaphors, written symbolic representations, language-based representations (spoken and written), pictorial or graphic representations (these could include graphic or models that can be manipulated).

Students may sometimes use their bodies to embody certain representations, yet the LTM does not explicitly include this form of representation in its model. The absence of embodied representations and gestures in the LTM suggests that representations in this framework are based on static models as could be the case in pen/paper activities. Meantime, in robotics and CT environments some tools can be dynamic involving the movement of objects. These movements can lead to the generation of representations that may take the form of gestures and embodied practices. Embodied practices are also valid representational media to observe the operationalization of CT activities and processes, especially for young learners without programming experience (Shokeen et al., 2020).

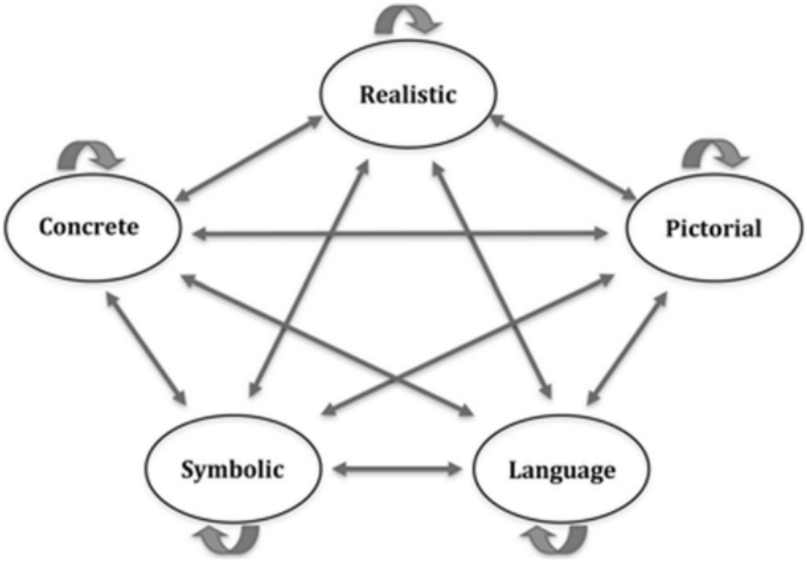


Figure 1The Lesh Transition Model

Multiple Representations

Given the highly multimodal nature of CT robotics programming environments, one would think of problem-solving in these environments from a conceptual point of view as completely ill-structured. Students in these environments are tasked with formulating problems in ways that can be solved both in the physical and virtual using computers and a programming language (F. R. Sullivan & Heffernan, 2016). These students need to be competent in navigating representations in virtual and physical contexts (Moore et al., 2020). Physical representations of objects are inherently common in robotics activities. Not only is there a need for students to have the skills to navigate physical representations, but they also need to have programming skills and language to instruct the robots to perform the physical tasks (Bers, 2005; Chen et al., 2017; Flannery & Bers, 2013; Moore et al., 2020; Sáez López et al., 2020). The tasks in these environments require students to translate across multiple representations, and

then communicate their solutions with their peers (Bers, 2018). There is therefore a multimodal interaction between representations in both the physical (3D) and virtual (2D) dimensions through which students must simultaneously navigate to get their tasks completed (Francis et al, 2016).

CT in robotics environments is not only limited to programming robots and is not just limited to problem-solving or completing tasks designed for the robots. Understanding and practicing CT skills require students to communicate and express real-world problems in ways that a computer or a machine can execute (Wing, 2016). All of this requires a certain level of fluency in communication among peers, the activity environment, and the computer (Papert, 1980; Bers, 2010). Because of the multimodal nature of the environment, communication cannot be uniform across the whole spectrum of activities. Students learn to develop and use different forms of representations to help them navigate their activities (Moore et al., 2020). These representations are sometimes constructed in situ and learners may develop language and symbols to help them navigate representations (Alibali and Nathan 2012). This movement between different forms of representations can be closely tied to what we may sometimes observe as CT skills (Francis et al, 2016). This strong bond between CT and representations and how the representations are generated is what I hope to understand in detail in this study.

Even though within this body of research there is a lack of consensus over children's skills at both representing spatial relations and symbolizing space, some constructivists theorists have built on Piaget's work to argue that there is a relationship between developmental progression, spatial and symbolic representations (Blaut et al,

2003; Piaget et al, 1956 and 1960). These theorists' work has served as building blocks to the literature about non-verbal forms of multimodal communication (Kress et al, 2001).

Mapping and cartography using digital tools have been studied in elementary grades to understand children's conceptualization of symbols and autonomous learning using technology. Map reading and map making are part of the elementary school curriculum since 1989. In these classes, children are encouraged to use Geographic Information Systems (GIS) software and computer-aided design techniques to design maps. This is all promoted by GIS education agencies (Ordinance survey, 2002; ESRI, 2003). Children's map-making strategies are a valid environment where we can see the authentic use of spatial representations and reasoning to support map drawing tasks. Map drawing using text and image tools in a software environment supports young children's understanding of geographical information systems (GIS) and allows children to interact with symbols that represent landmarks, fonts, texts lines, areas, and spaces they wish to explore.

Some studies argue that mapping is a "cultural universal" through which very young children can understand the language of location and symbolization (Blaut et al, 2003). This work has some spatial reasoning intersections with young children relying on symbols to accomplish mapping tasks in that some of the basic sets of variables learners must explore are like those a cartographer draws upon to represent information from their perceptions of the real world. These variables are shape, size, orientation (direction), pattern, color, or shade. These visual variables are the basic building blocks of how students engage in environmental mapping and representation of their data and

how this data interacts with their environment. Sometimes these symbols can be represented based on their levels of abstraction. Some of the models and activities students interact with in the Sphero.Math activities take the form of maps drawn on large sheets of paper, just like cartography maps. This work will analyze observations on how students navigate activity spaces, how they map/navigate these spaces using robots, and the kinds of symbols and representations they use to accomplish tasks.

Representations Manipulatives, and Symbols

Studies on manipulatives as symbols have reported that the use of concrete objects does not necessarily translate to children's understanding of symbolic-referent relationships (Uttal et al., 1997) The dual representation hypothesis posits that the concreteness of a model rather makes it difficult to be used as a symbol because children must first establish the relationship between the object and its referent to be able to use it as a symbolic representation. Mathematics literature reports that introducing manipulatives in mathematics classes can help young children understand abstract concepts and the written symbols used to represent the concepts (Uttal et al., 1997). However, several studies have also argued it is possible learners fail in making these connections between manipulatives and mathematics expressions (Bruner, 1966; Fuson & Briars, 1990; Hiebert & Carpenter, 1992; Resnick & Omanson, 1987).

There is therefore a learning curve involved in the establishment of a symbol-referent relation (Resnick & Omanson, 1987). Manipulatives must therefore be carefully chosen and crafted in ways that they can facilitate children's understanding of the relation between the manipulative and the concept they are trying to understand. Uttal et al, (1997) suggest that some kinds of manipulatives may be more effective in

helping children gain mathematical knowledge than others depending on how they are designed. For example, the Dienes Blocks have reported successful use of manipulative systems to relate to their referent in terms of units that correspond to the base ten number system by having blocks that represent ones, tens, hundreds (Fuson & Brias, 1990).

Computational manipulatives, on the other hand, have been reported to be very successful learning tools through their ability to provide immediate feedback and emergence using dual modes of representation (2D representations of the programs on screens and 3D execution of the program in the physical world) (Resnick & Silverman, 2005; A. Sullivan & Umaschi, 2018). Computational manipulatives are defined by Resnick et al, (1998) as manipulatives that are designed to possess computational capabilities; for example, Lego Mindstorm kits, Sphero robots, programmable beads, bee bots, and other computer-embedded toys. According to Sullivan and Heffernan (2016), the principles of constructionist learning theories to which computational manipulatives can relate are that (a) learning can occur through embodied practices (b) collaborations and discussions among peers during interactions can promote learning (c) the immediate feedback mechanism features (unique to computational manipulatives) can stimulate reflection, debugging and stimulate immediate actions during problem solving activities thereby promoting learning to happen through self-guided thought of the learners.

In addition to the learning capabilities manipulatives provide, the design of manipulatives can help young children connect and establish a symbol-referent relationship in ways that can help them understand abstract concepts in each domain

(A. Sullivan & Umaschi, 2018). For example, Cuisenaire rods are used in early mathematics learning to support young learners' conceptual understanding of mathematics (Manches & O' Maley, 2012). These rods are designed with specific relational meanings such as color, size, and shape purposely intended to lower the floor for children to understand the conceptual understanding between the manipulative and the target domain. Learning can therefore happen through interactions with manipulatives by initiating analogical reasoning, gestures, and consequently embodied cognition in learners (Fishkin, 2004; Manches & O'Malley, 2012). Seen from this perspective, it can therefore be assumed that the design of manipulatives can play an important role in the enhancement of symbol-referent relationships for young children, by lowering the floor for representations to emerge and be easily navigated.

Computational manipulatives can also be differentiated from other manipulatives in that they can have agency and be interactive (e.g., artificial intelligence agents such as robots). Secondly, they can be used to generate both tangible and virtual artifacts (Papert, 1993; Sullivan, 2011; Bih et al, 2020). Computational manipulatives when used as traditional manipulatives, that is used to teach other subjects and concepts in a target domain, tend to rely on analogical representations concepts (Sullivan and Heffernan, 2016). For example, Cuperman and Verner, (2013) in their study describe the use of robotics materials to model biological concepts in predatory plants. However, there are not many studies that have explored the use of robots as manipulatives to understand other subjects, making this area of research an important one to consider.

Models and Modeling

Most activities and tasks using manipulatives involving young learners are framed using real-world models or storylines. For example, to engage kindergarteners to program using basic sequencing skills, Strawhacker & Bers, (2015), designed a robotics curriculum that was integrated into the core science kindergarten curriculum that focuses on animal habitats and characteristics. Some activities include children building and programming a robotic animal in its habitat and children listening to stories about a robot and then individually creating a program for the robot on paper using a creative hybrid environment for robotics programming (CHERP) icons. One such stories was about the bus from the children's song ''Wheels on the Bus". These activities modeled the concept of sequences or things happening in a repeated cycle, in ways that kindergarteners can understand the concept of sequencing using real-world experiences.

Lesh and Doerr, (2003) define models as "conceptual systems (consisting of elements, relations, operations, and rules governing interactions) that are used to construct, describe, or explain the behaviors of other systems - perhaps so that the other system can be manipulated or predicted intelligently. For example, a mathematical model focuses on "structural characteristics rather than, the physical characteristics of the system" (Lesh and Doerr, 2003, P.10). Just as the word "conceptual construction" was used by Piaget, (1970) as his epistemology and model to describe knowledge acquisition by children, Lesh and Doerr have gone further to distinguish two meanings of the term "model" in learning spaces. The first term, "used as an adjective that is synonymous with the word "ideal" as in "model classroom" (Lesh and Doerr, 2003,

P.9). The second meaning, Lesh and Doerr provide, relies on Piaget's research on developing mathematical reasoning, models developed to describe conceptual systems he believed students relied on to understand mathematical experiences (Piaget, 1970).

There is therefore a difference between conceptual systems children use and the models researchers develop to describe children's thinking. My research will specifically look at the models children develop explicitly to construct, describe and explain concepts they encounter during their CT activities as they program robots. Teachers can therefore not always conveniently download skills, ways of thinking, attitudes, beliefs, and feelings, creativity, ways of knowing and doing from teacher to student (Lesh et al., 2003). These skills are therefore incumbent on the learner to develop by other means such as interactions, collaborations, imitation, peer observation etc., thus the importance of designing environments for the observation and understanding of how these skills are developed. This work will also contribute to the literature of the learning sciences that will consider digital models ascribed to children and not those researchers or teachers have preconceived as part of conceptual systems they can use to attain learning goals.

This chapter reviewed the literature on CT, representations, manipulatives, symbols, educational robotics, and programming robotics in early-grade classrooms. This dissertation lives at the intersection of representations, CT, and educational robotics. Given the centrality of representations in this study, I use the Lesh Translation Model (LTM), and Gestures as Simulated Action (GSA) to operationalize representations and representational gestures. In the next chapter, I describe the methodological approach I use to understand the nature of representations and CT

interactions learners engage in during robotics tasks. I present the design of the study, data collection strategies, and data analysis methods.

Chapter 3: Methodology

Introduction

My first research seeks to understand the nature of representations and expressions young learners engage in as they program robots in mathematics classrooms, and how their interactions with the learning environment contribute to successful problem-solving in CT tasks. In my second research question, I seek to understand how the tools and design of the robot and its programming interface support CT engagement for young learners. To address these research questions; I adopt a qualitative methodological approach to understand the nature of interactions that lead to representations and how the environment supports learners to navigate tasks in a fourth-grade mathematics classroom. In this chapter, I introduce the Sphero.Math curriculum (Appendix A) is designed to be used in this study by researchers from the project, teachers, and collaborators. I also present the study design, data collection strategies, and the school setting from which the data is being collected.

The Sphero.Math Project

The Sphero.Math project is a research project developed through a research practitioner partnership (RPP) (Coburn et al., 2013). This partnership is developed in collaboration with the District of Columbia Public Schools (DCPS) under the cornerstones project. Cornerstones are listed by DCPS as high-quality powerful lessons that students engage in through the DCPS units of study. Students enrolled in DCPS schools get to experience Cornerstones across grade levels and content areas including

ELA, math, science, social studies, art, health, P.E and world languages. Cornerstones lessons are taught as part of the DCPS curriculum.

Apart from being a part of the DCPS cornerstone project, the Sphero.Math project seeks to address issues of access, equity, and opportunity by studying the implementation of a fourth-grade mathematics curriculum intended to teach learners CT practices and mathematical concepts using Sphero robots (spherical programmable robots). Sphero.Math also seeks to understand the feasibility of integrating CT in mathematics classrooms and how this cross-disciplinary integration can support student's understanding of CT while deepening their understanding of mathematical concepts. This project was designed around the needs and resources the DCPS provided. Additionally, the Sphero.Math curriculum was designed through collaborations between researchers working on the project, a technology specialist, an instructional coach, and 4th grade mathematics teachers from DCPS.

The Sphero.Math curriculum contains a series of 15 lessons, with each lesson building off the other with levels of complexity of tasks and skill acquisition increasing over time. The Sphero.Math lessons run within the time allocated for mathematics and computer science lessons. Each of the Sphero.Math lessons are structured into three parts:

1. Engage: This is framing the activity and creating a real-world model around the activity. By framing the activity around real-world contexts, learners get to visualize/understand/connect to the problem the activity is trying to address.
2. Code: In this part, students get to write code, run the code, and refine their programs to make sure their codes run properly.

3. Debrief: During this stage, students review and discuss their programs. They share their experiences about what worked and what did not work as they tried to complete their tasks.

The Sphero.Math lessons include PowerPoint slides and sample codes to help the teacher navigate through the lessons with ease. Objectives for each of the lessons are clearly defined and structured to align with the 4th grade Common Core Mathematics Standards. The elements of CT the students will likely encounter as they program are carefully highlighted in the curriculum and sample program keys are provided for each activity. To make the curriculum engaging for young learners, the curriculum framed each activity around amusement park themes. For example, lesson 8 is framed to have a Sphero robot navigate a model of a monorail that moves visitors around the park from one attraction to the other (see Appendix A). The curriculum presented on Appendix presents the full Sphero.Math curriculum that features all 15 lessons framed around programming the Sphero robot to navigate activities.

In addition to understanding the content of the curriculum and the activities therein, the learners must write programs on their iPads to enable the Sphero robots to complete tasks. While writing these programs, learners tend to encounter concepts in mathematics and CT they must understand, use the resources the environment provides to develop useful skills and knowledge to help them complete their tasks. One of the goals of the Sphero.Math curriculum is to integrate CT and mathematics in mutually supportive ways and secondly, to achieve the design goal of providing empirical evidence of the learning opportunities CT and mathematics combined can offer students.

Mutual Supportiveness of CT and Mathematics

The Sphero.Math lessons are designed to align with DCPS math common core standards. The DCPS common core math standards are designed to address conceptual ideas around concepts in problem-solving, reasoning and proof, communication, representation, and connections. These standards define what students should understand and be able to do in their study of mathematics. The definition of understanding a student's ability to problem solve in this study is achieved by designing activities within the body of mathematics standards that respect "what is known about how students learn". The Sphero.Math curriculum supports such a research-based approach to how students learn. This is done by deliberately integrating CT concepts into mathematics lessons in ways that support learning progression and the development of student's mathematical knowledge, skill, and understanding in ways that can be developed and evaluated over time. Each Sphero.Math lessons have explicitly CT and common core mathematics standards learning objectives. In alignment with the DCPS practice, students were able to use CT tools designed in the Sphero.Math curriculum to model real-world problems, use appropriate tools such as graphs, charts, concrete models, rulers, protractors, tablets, the Sphero robot, and its corresponding software interface to explore and deepen their understanding of various concepts we will explore in this study.

The Sphero Robot

The Sphero robot (figure 3a) is a spherical robot that can be controlled through programming a tablet or smartphone (figure 3b). The Sphero robot programming

interface (figure 3b) provides different types of programming environments that can provide intermediate to advanced coding experiences for different ages of learners. Learners can program the Sphero robot using the Sphero.edu virtual application, designed to be compatible with mobile devices and desktops. The Sphero robot is made of a durable plastic shell and has a built-in accelerometer, gyroscope, and magnetometer. The Sphero robot has sensors that allow the Sphero to track its orientation and movement in space. The Sphero also has a built-in LED light that can be used to create different visual effects. The Sphero Edu interface is built to offer a variety of coding activities and other interactive ways of engaging with the Sphero robot. The Sphero Edu app offers three modes for learning to code - Draw, Block, and Text.

The drive and draw functionalities are used in the initial lessons of the Sphero.Math curriculum to provide practice experience in driving and controlling the Sphero robot, serving as a precursor to engaging in programming activities. After each drive, the Sphero.edu app displays relevant data, such as the distance Sphero traveled and the shape of the figure the robot made so the learners can visualize the movements of the robot (figure 2).

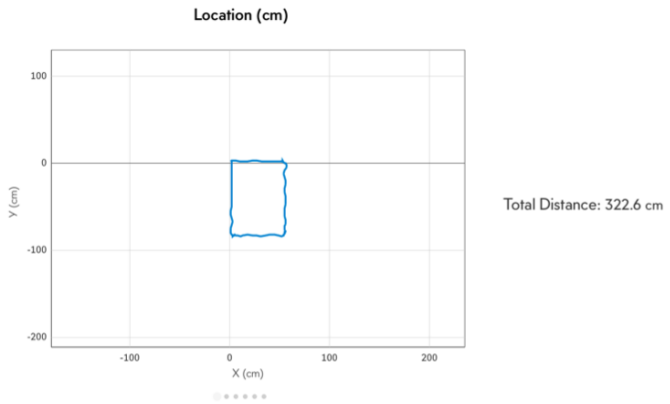


Figure 2 Display data showing the distance Sphero traveled.

The Sphero programming interface is also designed to program with block and text-based programming languages. The block-based programming interface (figure 3c) provided by the Sphero.edu app is like the Scratch programming environment. By selecting the "Blocks" option on the Sphero.edu app, students can access the block-based programming interface. Within this environment, students can start to use more advanced computer science concepts, such as loops, operators, functions, etc, rather than simply tracing a path with their fingers for the robot to navigate. A block-based programming environment is a graphical programming interface where users can create code by arranging visual blocks that represent code structures (Maloney et al., 2010). In a block-based programming environment, Syntax errors are eliminated by constraining users to connect blocks in syntactically correct ways, making sure that blocks fit together, and only logically valid combinations are possible. This visual approach reduces syntax errors and enhances the overall accessibility of programming, especially for beginners.

Students can build on some computer science skills they learned with the draw function to start to understand how to optimize the functioning of their robots. Students

can practice how to refactor codes, use loops wherever they identify repeated patterns, use functions and conditionals, and apply different codes to build block programs. The Sphero.edu block programming interface provides 11 block categories for students to use, including movements, lights, sounds, sensors, events, variables, functions, conditionals, and more. Students can explore this programming environment to learn how to program using block-based programming, refactoring, and how to write complete codes using different coding blocks.

The third coding method the Sphero.edu interface provides is the text-based programming environment. This is a more advanced programming environment, different from the foundational draw and block-based programming environments in that it uses JavaScript text. Learners did not specifically engage in text-based programming for this activity. Given that my research question 2 focuses on design scaffolds, the block-based programming interface is presented and analyzed in greeted depth in Chapter 5.

Due to the age range of the learners for this study, the curriculum only uses the block-based programming interface. The Sphero robot device is designed to be colorful and attractive (Figure 3a) with a spherical shape that children can easily cup in their hands and roll through surfaces in playful ways, making them very engaging tools for young learners. The Sphero robot comes in the form of a robotic ball with a clear shell showing circuits and wheels. The Sphero robot is designed with many programmable sensors and colorful lights. Sphero can connect to computational devices through infrared communication and its programmable sensors include a compass, light sensor,

gyroscope, accelerometer, motor encoders that can allow for manually aiming the Sphero robot or using the built-in compass to automatically aim and drive the robot.



(a) (b) (c)

Figure 3: The Sphero robot (a) iPad and programming interface (b) Sample program (c)

Sphero can be programmed to follow directions on a maze or to navigate paths. Sphero programs are represented as block-based scripts (see Figure 3c) that can use basic programming elements such as loops and conditionals, as well as commands to regulate the Sphero's movement or actions (e.g., roll, strobe). Additionally, these scripts can incorporate sensor data from the device, including metrics like distance traveled and speed. The fundamental movement command for the Sphero is the roll command (figure 4) which can take inputs for heading, speed, and time. The example shown on figure 4 is exemplified by a command that reads: "`roll 0° at 50 speed for 2s`."

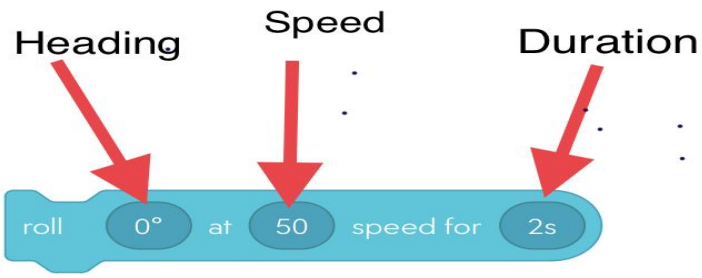


Figure 4 Sphero roll block.

Study Design

For this study, I collected video images from selected classrooms as students used the Sphero.Math curriculum in their respective classrooms under the supervision of their teachers. I took video images, observed, and interviewed students as they programmed the Sphero robots, and took notes. I conducted cognitive interviews with focal students as they worked through activities, and observed focal students' interactions during the activities, their conversations, gestures, and actions that lead to different forms of representations. At the end of each activity, I collected relevant artifacts students produced during the activity.

To understand the nature of interactions students, engage in as they program robots in mathematics classrooms using the Sphero.Math curriculum, I conducted a qualitative study after having observed and analyzed recorded videos of participants in their classroom setting as the teacher delivered lessons using the Sphero.Math curriculum. Using a combination of a computer or tablet, a Sphero robot, and access to the Sphero Edu app, which houses the block-based coding interface, students engaged in a series of multidisciplinary math lessons, using CT concepts, practical measurements, algebra, and geometry skills. Students used block-based programming

to program the Sphero to move from point A to point B by adjusting the time and speed inputs for the roll block (Figure 4). To accomplish this task, they employ measurement units and tools within a single system of measurement, honing their ability to perform precise tasks. Engaging with Sphero and the block-based coding environment allowed learners to develop a deeper understanding of CT concepts, such as algorithms, sequences, loops, decomposition, debugging, abstractions, functions, and conditionals, while also applying mathematical skills like measurements, algebraic thinking geometry, and spatial reasoning.

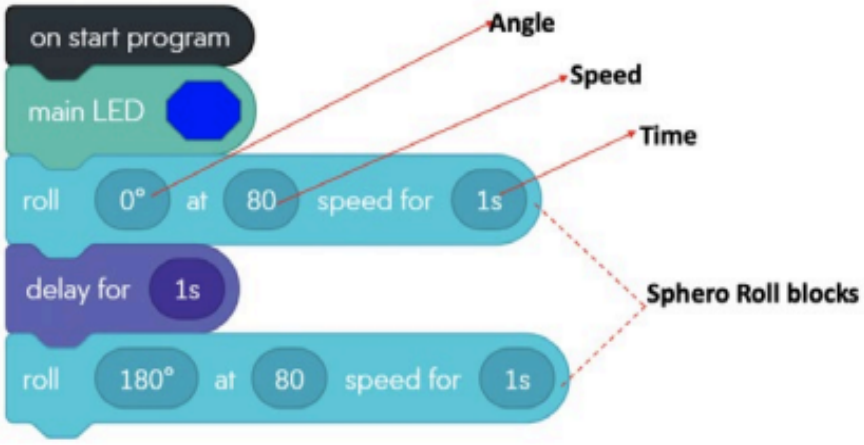


Figure 5: A Sphero Block program showing direction, speed, and time; (b) the software interface to select time speed, and direction.

Data Collection and Analysis

Settings and participants

This study was conducted in two fourth-grade mathematics classrooms located in urban public schools in the Mid-Atlantic region of the United States. As part of an

RPP, several professional development trainings were organized for DCPS mathematics, computer science and media lab teachers of the DCPS to use the Sphero.Math curriculum. The first school from which I collected data, has a total enrollment of 255 students with the following racial breakdown: 52% Black/African American, 1% Asian, 45% Hispanic/Latino, 0% Native American/Alaskan Native, less than 0% Native Hawaiian/Other Pacific Islander, 2% White non-Hispanic, 0% Multiracial. The second school has a total enrollment of 433 students with the following racial breakdown: 16% Black/African American, 0% Asian, 64% Hispanic/Latino, 0% Native American/Alaskan Native, less than 0% Native Hawaiian/Other Pacific Islander, 0% White non-Hispanic, 7% Multiracial. In total, 31 students from the first school and 42 students from the second school agreed to be recorded during activities, resulting in a combined total of 73 consenting students.

Classroom teachers selected and recommended focal students as participants for this study. I worked with two teachers who each managed two Sphero lesson classrooms. The teachers organized learners into groups to work on Sphero activities. The teachers designated focal groups, three focal groups from the first school (one from one class and two from the second class). The teacher from the second school designated one focal group from each class. The teachers did not clearly state why they proposed the focal students they chose in their respective classes. This study included a total of sixteen students, organized into five focal groups, drawn from four fourth-grade classes across the two selected schools. I observed and videoed these students during their interactions with the Sphero robots to understand how they navigated tasks, mathematical concepts, and aspects of CT elements, programmed the robot, and

discussed their perceptions about the robot's behavior. Two teachers were involved in this study and each of the lessons lasted 45 minutes. In both schools, the fourth grade was split into two classes,

Students selected as focal students were all consented; that is, they signed child assent forms and had their parents sign consent forms that permit for them to be videotaped during data collection and have access to the codes and materials they created during the activities.

Data Collection

The primary data source for this study is videos I collected during the Sphero.Math lessons, student artifacts, such as images of codes written in their computer science books, images of different stages of their codes, researcher field notes, and cognitive and clinical interviews. Due to the complexity and multimodal nature activity environment, I employed different data collection approaches to understand how young learners program robots in mathematics activities.

The first approach I employed was a task-based interview approach (Goldin, 2000). I asked questions about the tasks, as I saw actions that emerged (See Appendix C), I asked questions to understand the choice of decisions or actions carried out in certain instances. This approach is considered a form of a clinical interview (Clement 2000; Ginsburg 1981; Hunting 1997).

The second approach I used to collect data was through video recording using video cameras – head-mounted video cameras worn by the students, iPad and iPhone cameras to take close-shot images of the activities, and a video camera set up in the classroom to capture classroom activities. The use of different cameras gave different

perspectives of the interactions during the activities from different camera views. At the end of each lesson, I conducted short cognitive interviews to find out from students about their reflections on their activities.

Thirdly, I took notes of things I noticed as students worked and collected artifacts that they produced as they worked on their activities. After each data collection session, I also kept a log of field notes of important happenings I may not have caught on camera during the activity. I collected data from all four classrooms, from which I focused on at least one focal group of students. I collected data from a total of sixteen students (five focal groups) from four fourth-grade classes. For the first school, I collected a total of 1370.215 minutes (22.837 hours) of video and for the second school, a total of 911.472 minutes (15.1912 hours) of video was collected.

I conducted brief interviews (see Appendix B) with the focal students by asking them questions about the activity and what their experiences were during the process of accomplishing the tasks. I asked questions about the challenges they faced as they problem-solved and how they overcame the challenges. I also asked questions to understand if they had a full understanding of the respective programming blocks they used. Some questions were semi-structured, based on decisions or actions the students made as they worked through the activity. Questions suggested on the interview protocol in Appendix B are therefore not suggestive of all the questions I asked the students during the entire study. Most questions were rather generated as a function of the nature of their interactions to understand the actions and decisions they make as they program the Sphero robot. These questions are captured in the vignettes I present in the findings section.

Data Analysis Strategies

I used qualitative research methods to analyze my data. As I collected video data in classrooms, I noted particularly generative instances in my researcher notes for potential further review and analysis. After classroom data collection was completed, I viewed videos to select clips with episodes that showed learners producing and navigating representations during their interactions in ways that highlight how learners interact with the learning environment and its tools. Clips of generative instances were cut out for further analysis. Altogether, I reviewed a total of 2,281.687 minutes (38.028 hours) of video and cut out selected compelling video scripts that highlighted how learners used representations and interacted with the robot. These relevant clips were selected on the basis that they were clips with instances where I found learners use one of the five forms of representations to express CT practice. I looked out for specific instances when learners were expressing CT practices such as pattern recognition, algorithm design, problem decomposition, abstraction, and representational scaffolds that supported them engage in CT concepts or practices. I also watched out for instances when Math and CT were mutually supportive, looking out for the representational structures that supported the mutual supportiveness to occur. Instances, when learners engaged in CT concepts such as sequencing, using loops, debugging, iterating, and sharing their perspectives on either a problem or the solution, were particularly rich instances I found relevant for analysis.

In conducting my data analysis, I draw inspirations from interaction analysis (IA) as the primary methodological framework. This involved employing coarse-grained observations, focusing on specific interactions within the programming

environment to glean insights into the interactions learners engaged in the learning environment with the learning tools. I used a higher-level, coarse-grain observational approach, to analyze the "in use" and "in action" or "in practice" perspectives of cognition and learning (Hall & Stevens, 2015). Hall and Stevens (2015) further describe AI methods as "leaning towards a sense of natural occurring events in settings such as classrooms, workplaces, museums, family homes, organized by and for participants to capture a natural perspective of the environment". This qualitative analysis approach is better suited to answer my research question by looking at the cognitive-situative nature of the research question.

I used repeated analysis of the data, providing opportunities for alternative interpretations of the data with supporting evidence either in the form of transcripts or actual recordings for publication to readers (Sacks, 1995; Atkinson and Heritage, 1984). The multimodal nature of the Sphero.Math activities naturally produce different forms of data streams, requiring meticulously careful and sometimes repeated observation of the data to make sense of the connections; for example, moments, when the students expressed forms of knowledge during the activity either through verbal expressions, collaborations with peers, written out artifacts, information entered into their iPad in form of programming or on their notebooks and even gestures they make as they perform the activity, could have a combination of several things happening at the same time. This data collected in a naturally occurring way, captured events the learners engaged with without the researcher acting as a strong agent or intervening in those events. I did not control any variables in those events. I selected generative videos

worth a total of 320.82 minutes (5.347 hours) and from the second school, I selected 126 minutes (2.1 hours) of videos for transcription.

The data for this study was analyzed using the Lesh Transition Model (Lesh and Doerr, 2003) and GSA framework by Hostetter and Alibali, as an analytic lens, to understand the five forms of representations and interactions described below. These five forms of representations/interactions are:

Concrete Representations: Physical robot which, already by design, offers a concrete representation for hands-on experience and embodied learning just by touching.

Language representation: language representations used to create algorithms and control the robot. Verbal explanations between group/team members to facilitate communication, collaboration, and reflection.

Graphical representations: Grid based map serving as the maze.

Symbolic representations: Symbols to represent or convey information, concepts, or entities, abstract, arbitrary representations that carry meaning within a particular context or system.

Embodied representations: Embodied experiences with the robot and the learning environment.

For my second research question, which seeks to understand how young learners interact with the Sphero robot in order to express CT ideas, I analyze different moments when the students expressed different forms of representational knowledge during the activity either through verbal expressions, collaborations with peers, written out artifacts, information entered into their iPad in form of programming or on their

notebooks and even gestures they made as they performed the activity. I did not control any variables in those events and settings.

Analytic Approach

I used analytical memos for video data and interpretation. I used the interaction analysis framework to support repeated analysis of video data, and to interpret findings in the form of transcripts from video recordings as they occurred in their natural formats. I used IA methods to categorize data based on the codes that emerged, using at least two rounds of coding, and analyzed the data using the Lesh Transition Model (Leash and Doerr, 2003).

After collecting all the data, I analyzed the data in 2 phases. Phase 1 of my data analysis was inductive; I watched all the videos I collected from two schools. I open coded all the videos selected into transcripts, reviewed the transcripts, selected those with compelling use of representations and interactions with the programming interface in meaningful ways (see Appendix C). Phase 2 involved thematically coding the data (Saldana, 2015). I developed a codebook structured around two sets of themes, aligning them with the organization of transcripts in response to the research questions. I analyzed actions, quotes and interactions that fit into forms of representations using the Lesh Transition Model and interactions that showed evidence of the various ways in which the Sphero robot's design and features contribute to supporting young learners' engagement, problem-solving abilities, and skill development in CT environments (Resnick et al., 2005). To answer the first research question, I identified and aligned themes around forms of representations. For the second research question, I remained focused on the design aspects of the robot and the programming environments and how

these aspects can support learners' interactions. I organize these findings in the findings section. Exemplar quotes are used throughout the findings to demonstrate the identified themes.

I analyze the data on CT and problem solving by relating and matching them to CT concepts and practices defined by the Sphero.Math project guidelines (see table 3.1). Instances where learners used computational thinking practices, concepts and perspectives to explore and employ mathematical ideas and express any of the following 4 CT concepts (Pattern Recognition, Abstraction, Decomposition, and algorithms) as defined in table 4 below, will be coded as instances when CT is expressed during problem-solving. Such instances will be categorized under the four CT components identified to be observed in the Sphero.Math curriculum (see appendix A).

Table 4. Sphero.Math project CT component definition and examples

CT Component	Definition	Example
Pattern Recognition	Identifying parts of a problem that repeat. Repeating steps/concepts can often be performed by a computer	Seeing that a series of steps are repeated to measure a distance, so putting those steps inside a looping block (e.g. repeat)
Abstraction	Creating generalizations that are meaningful/useful or identify the essential parts of a problem and re-representing the problem including only the relevant parts	Being given the challenge of finding the perimeter of a specific rectangle but then writing a program that can calculate the perimeter of any rectangle
Decomposition	Breaking a problem down into smaller parts (either sub-problems or specific aspects of a problem that can be easily mapped to a computational solution)	Breaking down a complex shape into a series of line segments and turns which map easily onto programming commands in the Sphero
Algorithms	Defining a sequences of steps that can be followed to achieve some desired outcome	Defining a series of steps that can be executed by the Sphero to find the area of your classroom

Chapter 4: Features of the Sphero Robot and its Accompanying Programming Environment that Facilitate Students' Understanding of CT Knowledge

Introduction of Sphero Robot as a CT tool used for Sphero.Math Activities

To answer my first research question; "*What features of the Sphero robot and its accompanying programming environments facilitate students' understanding of CT knowledge?"* I explored aspects of the Sphero robot and its programming environment that supported students' understanding of CT knowledge. I provide vignettes with evidence that supports the idea that certain design features of the Sphero robot can facilitate young learners' interactions with CT knowledge. This chapter is organized to offer a comprehensive overview of the Sphero robot and its programming environment. Subsequently, vignettes are presented to illustrate how certain features of the Sphero robot served as scaffolds that facilitated interactions for learners. In this chapter, I present evidence of how certain features of the Sphero contribute to supporting students' understanding of CT concepts.

The Sphero robot, as described in Chapter 3, is a spherical, app-controlled device that offers a hands-on, engaging way for students to learn the fundamental concepts and principles of CT. The Sphero robot, as observed in my analysis, can be described as an interactive learning tool that allows students to interact with various

dimensions of CT concepts and practices such as abstraction, algorithm design, decomposition, and pattern recognition through programming.

The Sphero robot is equipped with age-appropriate features tailored for diverse age groups and learning levels, aligning particularly well with the targeted audience for the Sphero.Math curriculum (4th graders). The Sphero.Math curriculum (see Appendix A) uses the PRADA framework to capture CT concepts and practices. These are practices that can help learners use computing to explore and employ mathematical ideas. The Sphero.Math working definition of CT has 4 main components are listed in table 3.1. In my analysis, I use the PRADA framework to operationalize CT.

Overview of Sphero Robot

The Sphero robot as described in greater detail in Chapter 3, is designed with certain features that make programming accessible for young learners. In this chapter, I outline key features of the Sphero robot that support young learners to program. The Sphero robot is approximately the size of a tennis ball and weighing around 2 ounces, is lightweight, facilitating easy movement and control. The spherical design of the Sphero allows for effortless rolling and control. Its smooth surface ensures a firm grip, and the robot's lightweight nature further enhances maneuverability. With adjustable speeds, the Sphero offers flexibility in control. Equipped with various lights and sounds, it can communicate information or create visual and auditory effects.

For this study, learners used the Sphero SPRK robot (figure 6), that comes with a light blue and glass-colored shell that makes it attractive to young children.

The Sphero robot comes with accessories such as an inductive charging base that can hold the Sphero during charge and provide a connecting port for the USB charging cable. The Sphero kit also comes with a physical protractor and a maze tape.

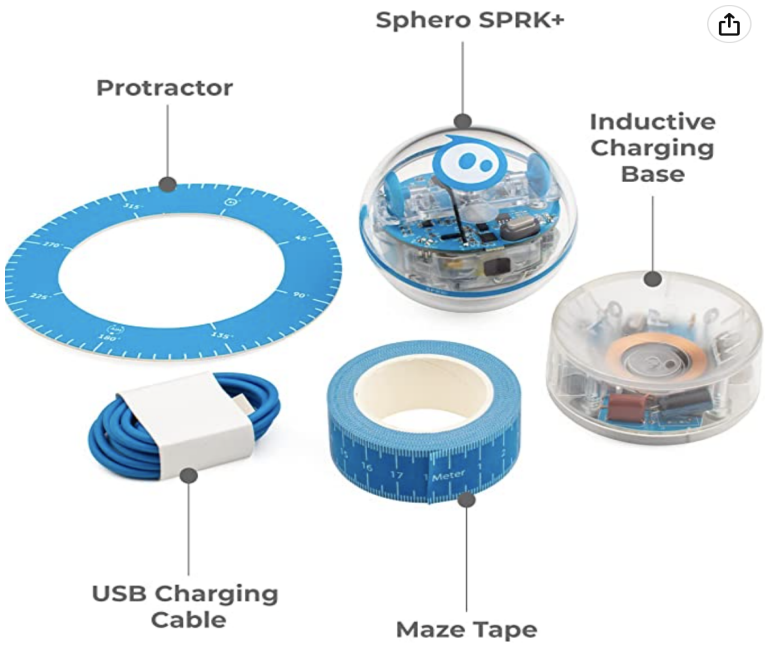


Figure 6: Sphero SPRK and its accessories.

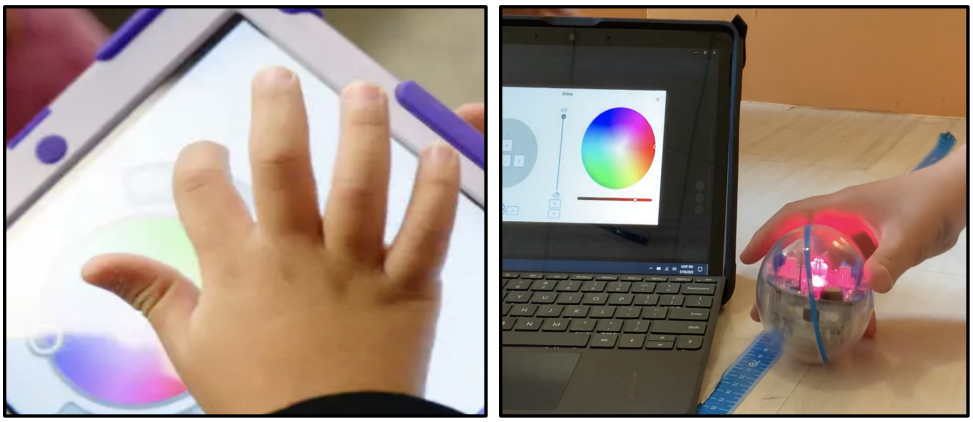
In the next section, I explore features of the Sphero robot, complemented by vignettes offering evidence of how certain features of the Sphero robot, assisted learners in solidifying their CT knowledge. I analyze excerpts from vignettes, capturing instances when learners engaged with these selected features.

Features of the Sphero Robot and Vignettes Supporting CT

Sphero Programming LEDs

The Sphero is designed to emit colored lights from its LEDs through its transparent plastic shell when a voltage is applied to the LEDs. The Sphero robot has

different types of LEDs that can be combined to make other colors through tactile touches on the color patch on the Sphero Edu app interface (figure 7b). To change lights on the Sphero robots, learners interact with the color panel (figure 7a) and select the color of their choice, with shades changing from lighter to darker shades using a virtual bar to the left of the color panel (figure 7b). The playful LEDs on the Sphero interface can change colors through tactile touch on the color wheel displayed on the screen, visible as they flash through Sphero's transparent shell (see Figure 7b). These LED lights can also be programmed as indicator lights to signify events, serving as a representation, or indicating the start position, action, or designed within the activity.



(a) (b)

Figure 7:Sphero LEDs on the robot and tactile interface panel on the Sphero app to select colors (b). The color wheel is used to change colors on the Sphero Edu app and Sphero with red color lights.

The Sphero robot's LEDs can be programmed to display different colors and patterns. In addition to helping learners understand CT concepts, the Sphero robot's LEDs can also be used to create engaging and interactive learning experiences. Learners can use LEDs to visualize and understand events about the activity. In Sphero.Math activity #8 titled "Monorail Stops," learners are challenged to program the Sphero robot to travel from the "Start" square to the "Stop" square, making intermittent stops at each "bus stop" and flashing its lights from yellow to green. I argue that programmable LEDs can be used to provide visual feedback for the learners.

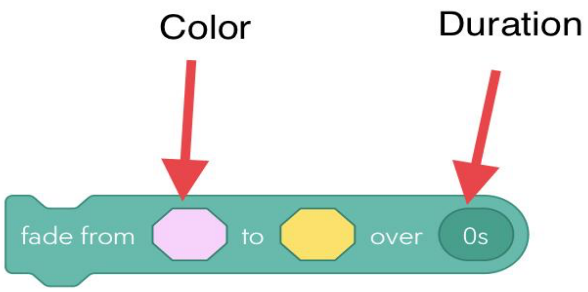
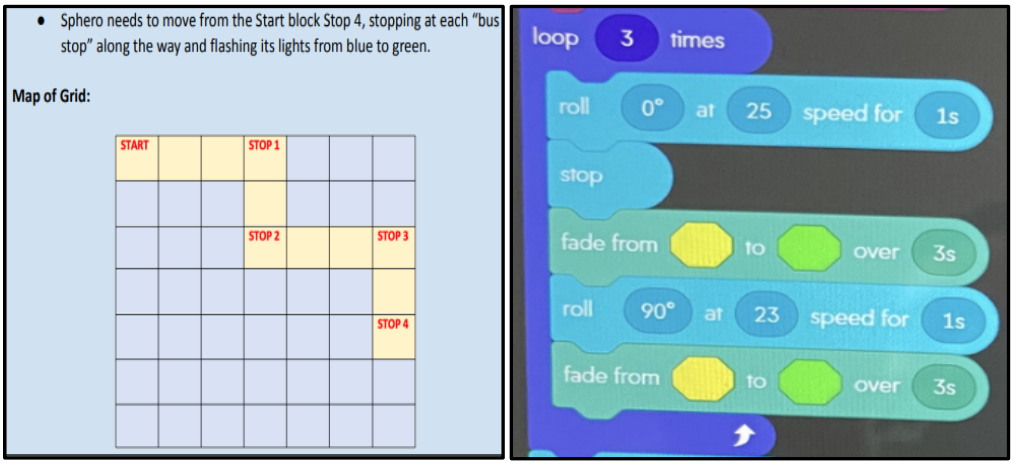


Figure 8: Sphero programmable LED block.

The programmable LED block (figure 8) enables users to select colors through tactile touches on color patches. This action opens up the color panel, offering an opportunity for color selection. Users can drag and drop this block into their program, adjusting parameters such as color duration to specify how long the chosen color(s) should persist.

Vignette for Sphero Programming LEDs

In this section, I present a vignette that foregrounds the LED feature of the Sphero robot, highlighting it as a feature that supports CT practice. The vignette presents three students, two boys and one girl, working on an activity to program monorail stops (figure 9a). Learners are tasked to make the Sphero move from start square to stop square flashing its lights at stops from yellow to green (figure 9b). The students in this activity make use of blink lights to visualize transitions in their programming activity, for example, the students program lights to indicate stop, wait, blink, speed.



(a) (b)

Figure 9: (a) Monorail stops activity grid with instructions on "flashing lights". (b) Programmable LED blocks integrated into a program.

Vignettes for Sphero Programmable LEDs

Student 1: I think we need speed and time. She [The teacher] said that and probably the distance and we need to find when to stop we also need blink two times.

Student 2: Blink from yellow to green

Student 1. This is what we need to do okay

Student 2: Yellow, green (while writing in her book)

Student 1: Speed and then heading then speed

Student 2: Three stop, stop for three seconds

Student 1: Heading... i just put heading

Student 2: We need to find how to blink, how to...(then reading the writing off her book) speed and time, stop, blink yellow to green stop for...(student 1 interjects)

Student 1: It just needs to be then put time

Student 2: ohh yeah, its first, heading speed and time then ermmm then stop for three seconds and then blink yellow to green...Then she moves over to the activity and moves her finger over the path on the maze

Student 1: There's no time on it? Help her find the block of time, he can't find it. speed time , blink yellow green.

Student 3: let's do blink

Student 1: Yah let;'s do blink first

Student 3: Yellow light?

Student 1: First is yellow then blink

Student 3: For three seconds? three right?

Student 1: Three seconds

Student 2: Do stop and wait for three seconds

Student 3: Stop and what?

//working on the tablet, entering the blocks and the values into the blocks as his teammates suggest

Student 2: Stop and wait for three seconds

Student 1: Stop and then wait for three seconds student 1. maybe we should stop and wait for 10 seconds instead so the people have enough time to get on

Vignette Analysis

I observed learners as they worked on programming the Sphero robot through a series of steps as designed by learners using blink LEDS and other actions to indicate events. The students were tasked with programming a Sphero robot to follow a set of instructions, wherein the Sphero's LEDs change colors to visually represent "Start" and "Stop" in the activity. The students had to make Sphero blink and wait, to indicate Start and Stop, while flashing its lights at stops from yellow to green. “Start” and “Stop” in this activity was visually represented by “blink” and “wait”. Students discussed elements they needed to include in their program to make the robot execute the intended sequence.

In the vignette presented above, student 1 says: *“I think we need speed and time she said that (referring to the teacher) and probably the distance and we need to find when to stop we also need blink two times”*. Student 1 by listing the steps, starts by identifying the key elements needed for the program, such as speed, time, distance, when the robot should stop and the number of times the robot would blink. The vignette indicates that the student recognizes blinking as part of the actions needed to execute the desired sequence. This evidence is supported by student 2 when he says, *“We need to find how to blink, how to...(then reading the writing from her book) speed and time, stop, blink yellow to green stop for…”* The requirement to blink required the students to wrestle with algorithms. This required students to approach the task systematically and logically by organizing it into a sequence of well-defined steps, breaking it down into smaller, manageable components to arrive at a solution. Other than programming speed and time into the solution, the blink lights are also

included as a resource for learners to visualize and understand part of the problem-solving process by connecting the activity to real-world contexts. As shown in the vignette, programmable LEDs were able to support CT practice in the following ways:

Decomposition: Student 1 mentions the need for the LEDs to blink two times. This means the number of blink times is important to meet a specific requirement. This was seen as two students having conversations and deciding which comes first; student 1 asks his peers "*There's no time on it? Help her find the block of time, he can't find it. speed time, blink yellow green"*; student 3 responds *"let's do blink"* Student 1 then confirms *"yah let's do blink first"...* Student 1 in confirming the order of sequence is decomposing the task with his peers. While breaking down the tasks, the students created opportunities to organize and structure various components of the activity to align the blink LED lights to events such as "Start", "Delay", and "Stop" as designed into the task. This is evidenced when student 3 asks *"Yellow light"?* And then student 1 confirms *"First is yellow then blink",* and student 3 further asks *"For three seconds? Three, right?"* providing clarity on the duration the LEDs will blink. Each colored light conveys information; for example, yellow to green blink signals that passengers can board the train. This vignette provides evidence that LED lights supported the students to decompose the task into manageable sequences.

Pattern Recognition: The students used the LED lights to get visual feedback that helped them connect to patterns and steps in the task. This is seen in the vignette as

student 2 recalled the need to change the LED color from yellow to green. Student 2 says "*oh yeah, it's first, heading speed and time then ermmm then stop for three seconds and then blink yellow to green...then she moves over to the activity and moves her finger over the path on the maze"*. This action demonstrates that the LEDs offer visual cues learners could rely on to delineate distinct stages of the sequence, thus helping the students to identify patterns within the activity. The LEDs provide visual feedback as seen in this vignette by providing immediate and intuitive feedback to understand sequences and consequently recognize patterns within the activity.

Sphero Taillight ("aim") LED

The Sphero's "aim" light is one of the most frequently used LEDs on the Sphero robot. When turned on, the "aim" taillight is visible dot light on the side of the Sphero robot (figure 10a). The Sphero's direction can be aimed using the aim function. When the "aim" light is set to face the programmer, the Sphero is set to heading 0 degrees (figure 10b).

Sphero robots are designed to operate within a geocentric reference system, which relies on a global reference frame derived from external sources such as GPS or a compass (i.e., turn north).This contrasts with an egocentric system, where the robot's perception and navigation are based on the robot's local frame of reference (i.e., turn right). Robots like Logo robots and other floor robots use egocentric systems for direction. Due to the geocentric nature of the Sphero robot, the direction/degree does not change when the heading changes on the program. Sphero robots are programmed to execute movements like moving forward (0 degrees), moving right (90 degrees),

moving left (180 degrees), and moving backward (360 degrees) using an internally designed compass system (figure 10b).

The act of positioning the robot's taillight to align with a forward movement is called "aiming" Sphero. The "aim" light on the Sphero robot is a small, blue light located on the side of the robot facing the programmers when the robot is placed in position (10a).

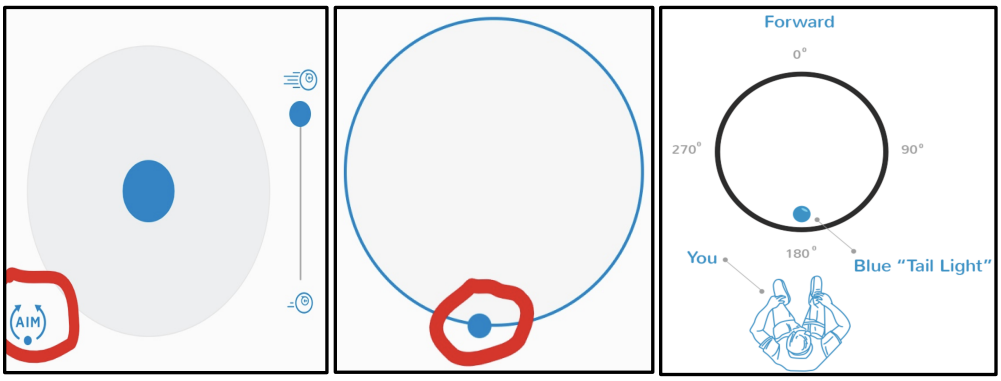


(a) (b)

Figure 10: (a) Aim (taillight) on Sphero robot (b) Heading set at 0 degrees on the Sphero interface.

The "aim" light is used to indicate the direction (heading) that the robot is facing (figure 10a). The "aim" light will flash blue when the robot is turned on. Once the robot has been paired with a device, the "aim" light remains solid blue. To "aim" the robot, students use their index finger in a tactile mode, to rotate the "aim" light on the tactile panel designed to set the "aim" light using tactile motions (figure 11a) until it points to the desired direction (figure 11b, 11c). The "aim" light defines where 0 degrees will be for the robot, so the direction the "aim" light is pointing is the direction the robot will roll when set to roll at 0 degrees. The "aim" light is an

essential tool for controlling the Sphero to align with the heading or the direction will travel.



(a) (b) (c)

Figure 11: (a) Tactile panel for "aiming" the Sphero taillight (b) Taillight (aim) indicator on Sphero interface (b) Position of programmer in alignment to the "aim" taillight (c).

Vignettes for Sphero Taillight ("aim") LED

In the vignette below, I present snippets of scenes where students in different instances across different activities use their bodies as a support tool to help them align the Sphero "aim" light to their desired heading. The students in this vignette engage in embodied practices to connect to the robot's aim lights to find orientations between the robot and the activity. In this study, I analyze embodied practices through the lens of literature on GSA (Hostetter & Alibali, 2010; Nathan et al., 2017) and embodied cognition theories (Barsalou, 1999; Glenberg, 1997; Lakoff, 2012), that argue abstractions are metaphorically grounded in embodied and situated knowledge. I present the "aim" light as a symbol to indicate the heading or the direction the robot will travel. In Chapter 5, I provide evidence that symbols serve as

representations for abstractions that simplify complex ideas. Engaging in embodied practices can help students connect visual cues, expressed through a sense of their body's position and movement, with abstract concepts, thereby enriching the learning experience (Weisberg & Newcombe, 2017). Additionally, embodied perspectives can create analogies that align abstractions with the empirical system in ways that can support learning (Wilson, 2002).

In the vignettes below, I present evidence of how learners use embodied perspectives to connect the "aim" light on the robot with different parts of their body.

Vignettes Foregrounding Sphero "aim" Taillight.

Student 1: So, this Sphero is supposed to go in a rectangle and end up back here, hang on hang on

(moving the Sphero in a roll position on the activity board and then placing his finger on the tail light to position the Sphero on the aim position his finger stays in that position for a few seconds.

Student 2: So, this is what you are going to do right, so you are going to look at this and tell us if anything is wrong and then you can add anything if you think anything needs to be added all right? So here's the pencil right there all right?

Student 2 runs the code while student one has his finger in place on the position of the taillight for about 30 seconds. The Sphero makes a rectangle and comes back to the original spot where Student 1 had his finger and then Student 1 says in a satisfactory tone, "pretty good!"

// Researcher observation: Student 1 sets the "aim" with her two feet set together to align on a straight line with the "aim" light.

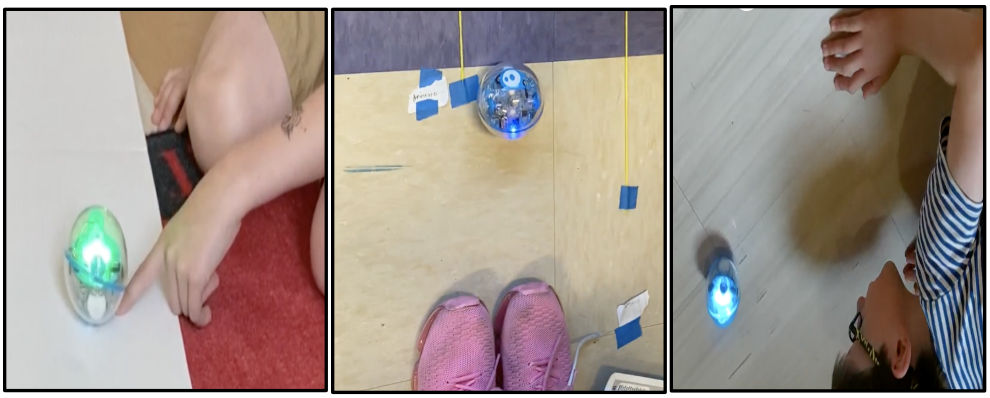
Researcher: Why are you putting your feet together?

Student 2: Because she said we have to aim it at our feet.

Student 1: If we aim it at our feet we're gonna get Sphero to roll straight.

(and then the virtual protractor pops up where she drags the arrow to the downward position).

Student 1: We also needed to stop over here first, (pointing at the screen on a position within the code)... "Oh we forgot roll, where's roll", and then pulls the roll block from the movements category.



(a) (b) (c)

Figure 12: Students use parts of their bodies to align and find orientation for the "aim" light on the Sphero. (a) Student uses his fingers to align to the "aim" light (b) Student uses her feet to align to the aim light (c) Student lowers his head to align his gaze with the taillight.

Vignette Analysis

In the vignette snippets above, the students use three different parts of their bodies (fingers, feet, and eyes Figure 12) to support aligning the "aim" light to make the robot move in the direction they desire. The student seen in Figure 12a uses his finger to align with the "aim" light. The researcher observed that *"the student moving the Sphero in a roll position on the activity board and then placing his finger on the taillight to position the sphero on the aim position his finger stays in that position for some seconds"*. By aligning his finger with the "aim" light, the student creates an analogy for the direction the robot is intended to move, thereby creating a pattern. This action involved abstracting the robot's "aim" light to synchronize with the position of his finger, ensuring precise movement in the desired direction.

The researcher observed student 1 set the "aim" light from her iPad with both feet set together to align the "aim" light on the Sphero robot. When asked why she had her feet together, student 2 responded: *"Because she (the teacher) said we have to aim it at our feet"*. Student 1 further explained that "*If we aim it at our feet, we're gonna get Sphero to roll straight".* Based on these respective analyses from the vignettes and images captured, I argue that the taillight called "aim" on the Sphero robot was able to support student's CT understanding in the following way:

Abstraction: According to Wing (2011, p. 20), abstraction serves several purposes: *defining patterns, generalizing from instances, and parameterization*. Wing further argues abstractions can *enable one object to represent multiple entities and captures shared essential properties among objects while concealing irrelevant differences*. The student's engagement with different body parts (fingers, feet, and eyes Figure 12) showcases how abstraction enables complex interactions through simplified means. By using multiple body parts to control the Sphero robot, students are effectively generalizing from instances, as each body part serves as an abstraction for a specific action (aligning the "aim" light to the direction the robot should roll).

Furthermore, I argue that the learner's use of multi-sensory engagement is the power of abstraction in allowing one object (the Sphero aim light) to offer opportunities for diverse interaction possibilities beyond traditional input methods (heading block). Engaging multiple senses can support students' thought processes that are rooted in their bodily experiences and influenced by interactions with the physical environment. Additionally, the design of visual cues like the "aim" light on

the robot supported the engagement of embodied practices that enabled the students to construct sensorimotor referents, acting as tools to contextualize abstractions such as directions the robot will travel within familiar geometric concepts they could readily understand and manipulate.

Sphero Virtual Protractor

The vignette I present is video excerpts from learners as they worked on lesson #11 titled “Triangles” in the Sphero.Math curriculum. This activity challenged students to write a program to draw an equilateral triangle. This lesson involves converting from the desired interior angle measurement to the exterior angle measurement required as input for Sphero's internal compass interpretation.

Success in this lesson is determined by students' ability to leverage their understanding of how Sphero interprets headings, allowing them to write a program that instructs Sphero to draw an equilateral triangle. This lesson is the second of a two-day lesson framed around Park Beautification that challenges students to write a program that draws an equilateral triangle.

In the vignette below, a pair of students interact with the Sphero.edu app using tools presented on the app to program a triangle tapped to the floor (figure 13).

Vignettes for Sphero Virtual Protractor

//Researcher’s observation: when the students changed the start position on the triangle taped to the floor, they succeeded in navigating two sides of the triangle, and were unable to make the Sphero navigate the third side. Faced with this challenge, student 1 starts walking on the triangle tapped to the floor, while visualizing the virtual protractor on the iPad to find the direction. He uses his body as an orientation to directionally find how the Sphero will move figure 13).

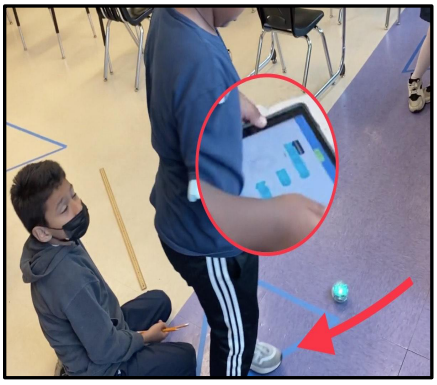
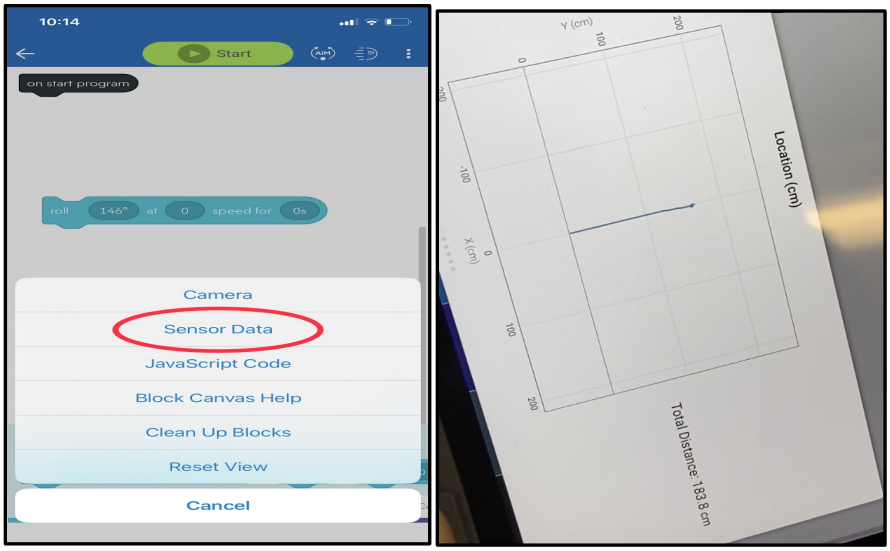


Figure 13: Student walking on the triangle with the iPad and visualizing the virtual protractor.

His body and the virtual protractor become the affordable tools he can rely on to map the activity on the ground, onto his body, and the virtual protractor. He frequently pauses to use a physical protractor to measure angles, possibly in the expectation that the protractor will yield specific data or a numerical value for input regarding direction or angle.

Student 1: So make it this way (using his hand to motion his group mate the direction the aim button should align to) so it could turn that way (again using hand gestures to indicate directions for the aim)

The students used the physical protractors to measure the angles and then relied on the virtual protractors to enter the angles at 61 degrees (figure 15). The students sometimes queried the sensor data to provide clearer visual image of the actual direction the robot rolled on the the cartesian coordinate plane (Figure 14b). The sensor data icon (Figure 14a) is a visual cue that students query to visualize recorded data about the robot's movement as captured by the robot's sensors.



(a) (b)

Figure 14: (a) Sensor data icon. (b) Sensor data students queried to see the direction their robot travelled.

And then one more time the student steps on the triangle tapped to the floor, walks on the of the sides of the triangle with his left hand in front of him, holding the ipad with his right hand, looking into the virtual protractor on the ipad, and changing the directions on the virtual protractor as he walks on the tape representing the triangle.

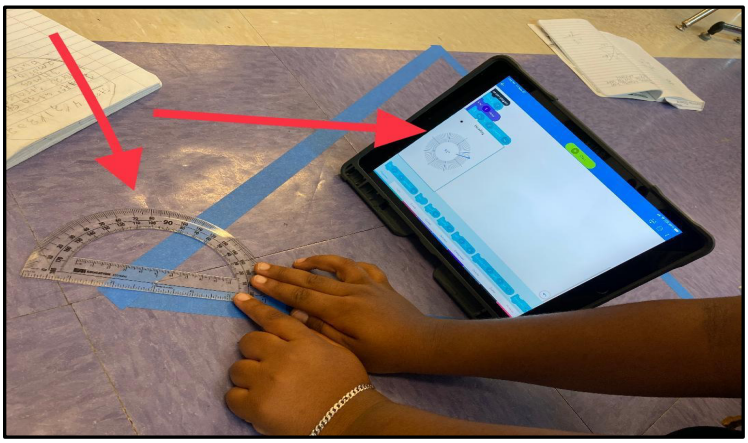


Figure 15: Student using the virtual protractor and physical protractor to support their understanding of angles on a triangle.

In the following vignette that I examine, I highlight the virtual protractor as a crucial tool that supported learners to convert the triangle's interior angle measurements into the exterior angle measurements necessary for input in Sphero's internal compass interpretation.

The virtual protractor on the Sphero.Edu app is designed to mimic the functionality of a physical protractor. The virtual protractor allows students to select angles using tactile movements to interact with the arrows to select angles (figure 15).

The use of arrows to represent angles on the virtual protractor involves indicating the direction or orientation of the angle (figure 16). The arrow typically points towards the direction in which the angle is measured, simultaneously displaying the direction and the value of the angle.

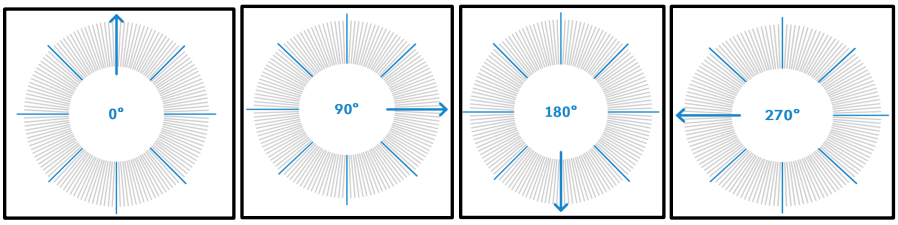
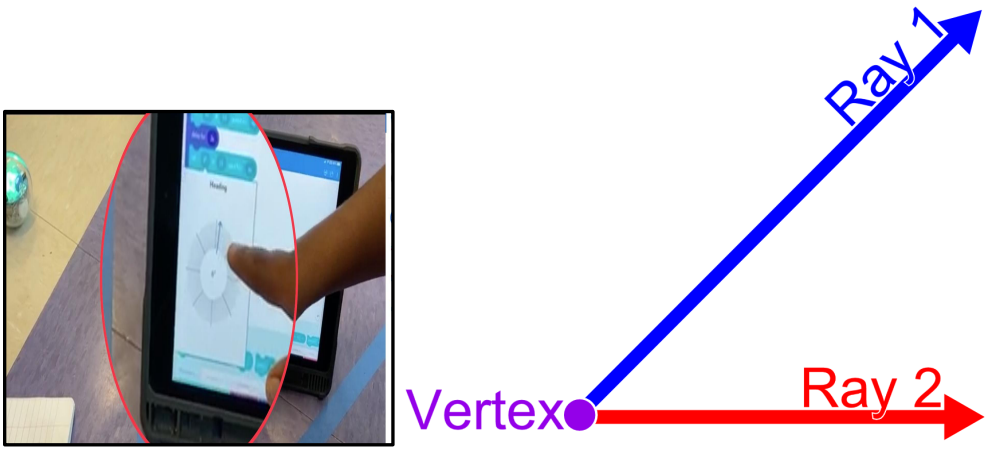


Figure 16: Arrow symbols within the virtual protractor representing various angles and directions indicating the robot's intended movement.



(a) (b)

Figure 17: Students use tactile movements to rotate the arrow to select angles.

The Sphero's virtual protractor is designed so that, the ray of the arrow extends from the center position on the sphere, to represent the heading or the intended direction of Sphero's movement. As shown in Figure 17b, an angle is defined as the geometric shape formed by two rays, known as the sides of the angle, sharing a common endpoint called the vertex.

Vignette Analysis on Sphero Virtual Protractor

The student uses the virtual protractor to help him embody the angles of the triangle. *When the students changed the start position and set the robot on the triangle taped to the floor, they succeeded in navigating two sides of the triangle, and were unable to make the Sphero navigate the third side. Faced with this challenge, student 1 starts walking on the triangle tapped to the floor, while visualizing the virtual protractor on the iPad to find the direction.* This vignette provides evidence that the virtual protractor on the app can support the enactment of embodied practice. The arrow and angles on the virtual protractor served as abstractions that simplified

students' understanding of angles and the robot's movement directions. The arrows on the virtual protractor facilitated embodied practices, allowing learners to connect visual cues and directions with their own body's position and movement on the triangle's sides. This connection enabled students to link their physical engagement with the abstract concept of angles, as proposed by the robot's internal compass system.

The Sphero app's visual representation of the protractor was a support tool for students to navigate angles. The visual aspect of the protractor provided scaffolds for students to form images and representations of the angles they encountered in the activity, thus supporting their ability to visualize and abstract the complex concept of angles using their bodies. The vignette shows how the student engages in deeper embodied actions using his hand and body together with the virtual protractor to find angles (figure 18)

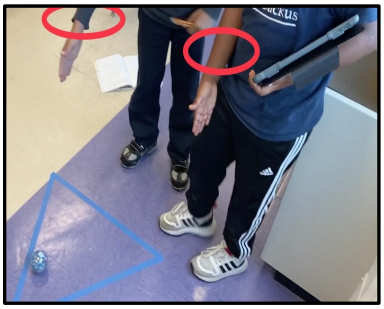


Figure 18: Students connect hand gestures to embodied practices to find the robot's directions.



Figure 19: Students connect hand gestures to embodied practices to find the robot's directions.

Figure 19 shows how the learner embodies the triangle path, as *he steps on the triangle taped to the floor, walks on the sides of the triangle with his right hand in front of him, holding the ipad with his left hand, looks into the virtual protractor on the ipad, and changing the directions on the virtual protractor as he walks on the tape representing the triangle (figure19).* The learner is using his body to map the direction the robot will travel on the path. By linking their body movements to the virtual protractor, students were able to bridge the gap between the physical triangle and the abstract concept of angles linked to the robot's movements and directions. This connection improved the student's understanding of the robot's roll direction and functioned as a debugging tool. It offered immediate feedback, enabling the student to modify and adjust angle values and, consequently, the robot's movement directions.

The design of the virtual protractor helped the students use the CT component of abstractions to overcome the complexity of the geocentric nature of the Sphero robot. The CT component of abstractions helped students create generalizable meaningful scaffolds through embodied practices in ways that helped them understand and represent the angles appropriately. This observation was consistent

across both schools, where teachers sometimes used embodiment to scaffold learners understanding of directions and angles (figure 20).

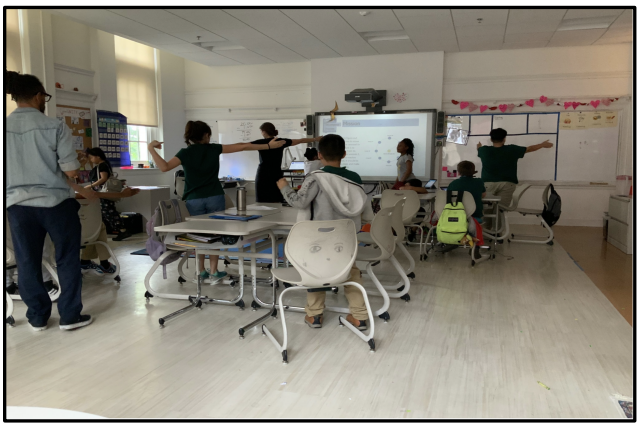


Figure 20: Students engage in embodied practice to understand the robot's directions.

In the next section, I explore how the design of the programming environment supported learners to explore CT knowledge. To program robots, students are usually tasked with converting the robot's physical movements into a symbolic system called programming. This entails planning the robot's actions, coding the instructions using a programming language, and assessing the outcomes of the programmed actions (Bers et al., 2019; Elkin et al., 2016). To attain successful results, students need to understand symbolic representations in computational tasks (Di Lieto et al., 2017; Kopcha et al., 2021). Symbols play an important role in facilitating learner's interactions with programming blocks and other symbols on the programming interface.

Sphero Robot Programming environment

As mentioned in Chapter 3, the Sphero programming language offers different ways to make the Sphero robot move. Students can either program the Sphero robot using one of the programming languages or the drive functionalities. Selected vignettes show instances when learners explored the block programming environment and how the design of the blocks and the environment supported their interactions.

Overview of Programming in Sphero Edu App

The Sphero Edu application serves as a platform for users to program the Sphero robot. It can also be used as a didactic tool for learners to acquire programming and CT skills. The Sphero.Edu application's interface comes with a user-friendly interface that offers visuals and interactions that support learners' engagement with CT concepts. Details of Sphero's programming interface are described in detail in Chapter 3. Below I present features of Sphero's programming interface that support learners' interactions.

Supported programming languages for Sphero Robot

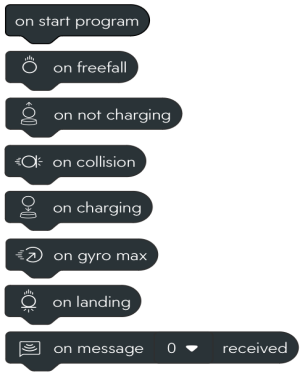
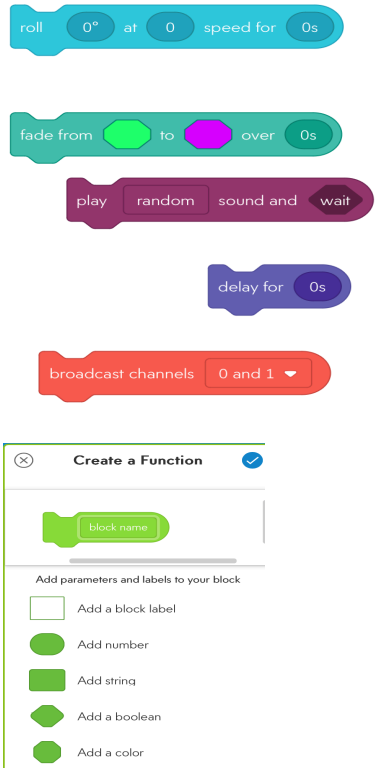
Block Mode: The Block mode introduces users to more advanced logical concepts via a block-based programming interface. It employs programming blocks, allowing users to visually script their commands and build their understanding of coding logic. It's a more intricate level of programming that goes beyond the basic tactile pads used in the drive draw mode. Block-based programming is a programming paradigm that uses visual elements or "blocks" to represent code concepts. Instead of writing code using traditional text-based syntax, users drag and drop visual blocks that contain

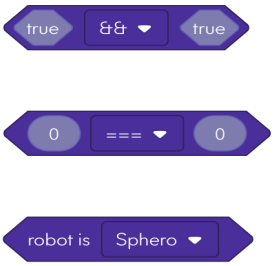
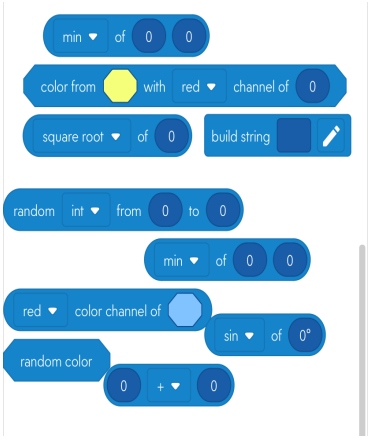
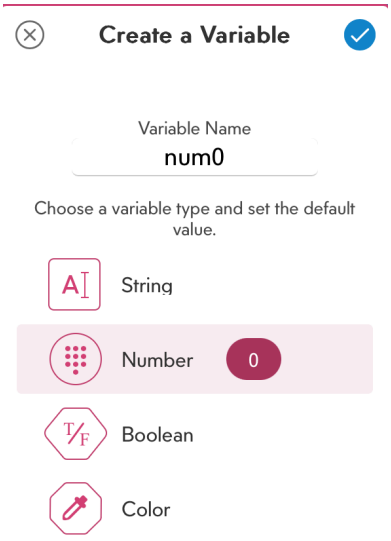
code instructions (Weintrop, 2019; Weintrop & Wilensky, 2015). Each block typically represents a specific command or operation, and users assemble these blocks in a sequence to create a program. The shape and color of programming phrases serve as visual cues, offering guidance on their assembly. If two programming phrases cannot be connected to form a valid sequence, the environment prevents them from snapping together, effectively avoiding syntax errors during program authoring (Lin & Weintrop, 2021). In block-based programming environments, block shapes, along with their combination rules, serve the function of syntax in a text-based language (Maloney et al., 2010).

Lin and Weintrop identify block shapes and colors as important design features that can support learners navigate the block programming environment. Blocks are puzzle-piece shapes used to create code by connecting to each other vertically by stacking them together in a jigsaw puzzle (Weintrop, 2019). Each block type has its own shape, color and a slot it connects with. Inspired by literature from the Scratch block programming environments, I provide descriptions for the blocks designed into the Sphero app editor, describing their shapes and colors in table 5 below.

Table 5. Features of Sphero Blocks.

Block Function/Type	Block Shape Description	Block Visual Representation and Color

Start block	The Sphero start blocks are blocks that start every program. They are shaped with a bump at the bottom so that blocks can be placed below them. There are eight start blocks in the Sphero app editor, seven of which are in the events category, one in the control category.	
Stack	Stack blocks in Scratch execute specific commands and are designed with a notch at the top and a bump on the bottom, facilitating placement above and below them. With a total of 44 Stack blocks in the movement, lights, sounds control and communications categories. These command blocks represent the most common block shape in the Sphero app editor. The function block takes the shape of a stack block and nests other parameters such as booleans, strings and numbers.	

Boolean	Boolean blocks serve as conditional elements, characterized by an elongated hexagonal shape. They can hold and reporting "true" or "false" values. These blocks are inserted into corresponding slots of other blocks with the same shape.	
Reporter	Reporter blocks serve as value carriers and have rounded edges. These blocks can hold numbers, strings, operators and reporting variables, there is theoretically an infinite number of reporter blocks that can be created for each variable and list. These blocks are inserted into corresponding holes of other blocks with the same shape.	 

Cap (End block)	The cap block marks the end of programs and have a shape featuring a notch at the top and a flat bottom, preventing the placement of blocks below them. In the Control category, one Cap block is available.	
Sensors	The Sphero robot is equipped with various sensors, including a gyroscope, accelerometer, and motor encoders. Programming Sphero sensors involves the use of pill-shaped sensor blocks, distinct from stacking coding blocks. Sensor blocks are nested inside other coding blocks, as they return numerical values. Any location suitable for a circular/pill-shaped block in the program corresponds to a numerical value. Sensor blocks are commonly employed in conditional and comparator blocks. The gyroscope sensor block for example measures how many degrees per second the Sphero robot spins.	

The Sphero block programming environment has a total of 11 built in categories with shapes and colors of different categories of blocks playing a crucial role in supporting young learners' understanding of programming icons and symbols. Blocks categories on the Sphero.Edu app contain a variety of blocks that can be used to control movements, lighting, sounds, controls, functions, operators, comparators, sensors, and other categories. For example, the movement blocks can be used to make the robot roll forward, backward, or turn left or right.

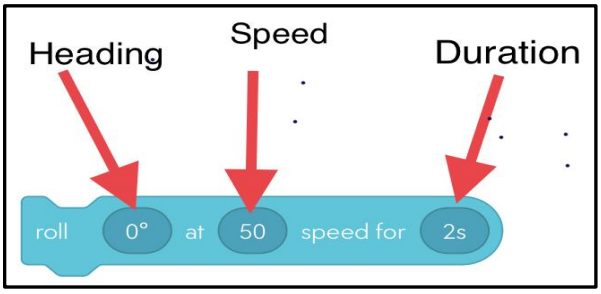
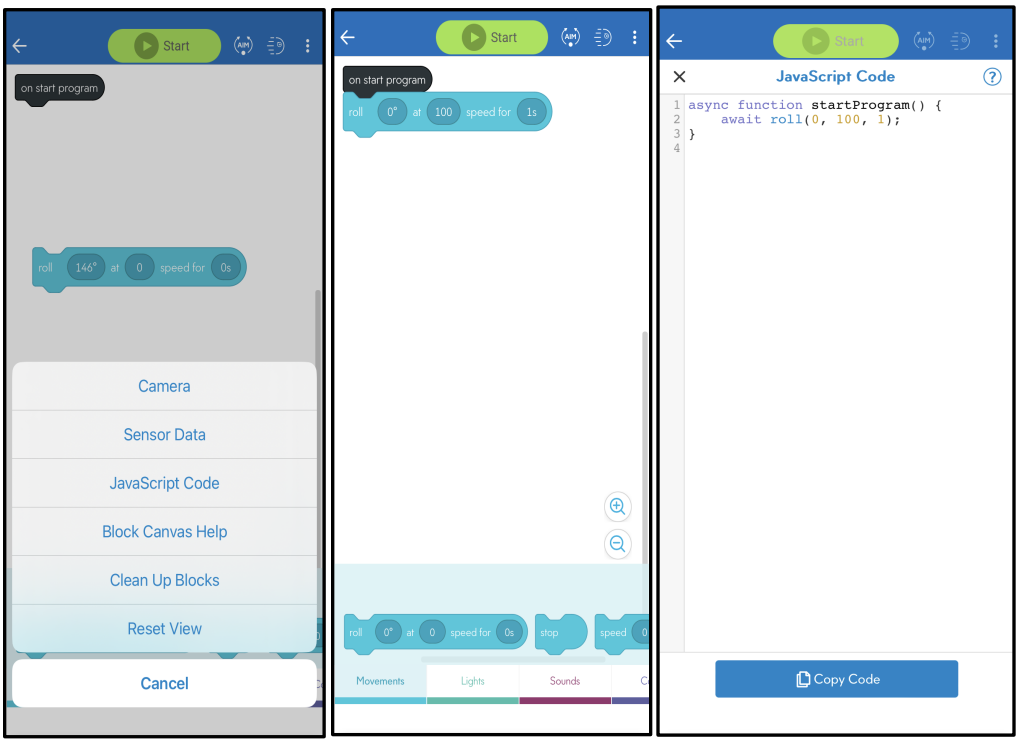


Figure 21: Sphero movement block, the "roll block".

The Sphero movement block also known as the "roll block" is an ocean blue colored block that has a notch at the top and a matching bump at the bottom that can be stacked with other blocks to create a program. The "roll block" allows learners to control the movement of the Sphero robot using parameters such as heading, speed, and duration (figure 21). Users can drag and drop this block into their program and customize it to instruct the Sphero on how to move. Once the blocks have been dragged onto the block canvas, students can connect them to create a program. To do this, they drag the output of one block into the input of another block (figure 22b). Once a program is created it can be run by clicking the "Start" button. The Sphero robot will execute the program and perform the actions specified. Sphero block

programming presents a low entry barrier for either a beginner or an experienced programmer. There is also a possibility to view the sensor data and JavaScript code for the blocks code written (figure 22c).



(a) (b) (c)

Figure 22: (a) Block programming interface showing the roll block with the possibility to view text code and sensor data (b) Blocks connected to create a program, (c) Sphero Block programming code changed to text JavaScript code.

Text Mode: For advanced programmers, the Sphero.Edu app offers a "Text" mode that opens opportunities for learners to explore the world of text programming. Here, users can write their own JavaScript to control the Sphero robots (figure 22c), allowing for a much broader scope of commands and actions that the robot can execute. This mode is ideal for learners already familiar with the basics of coding and

seeking to advance their skills. The text mode was not explored in the Sphero.Math curriculum hence was not analyzed in this study.

Sphero Block-Based Environment/Language

Sphero robots are equipped with various sensors, students can program to vocalize collected sensor data using the speak blocks. This involves constructing strings, sequences of symbols or characters, which can support the learners to debug programs. Sensor blocks, represented by features such as the accelerometer, gyroscope, and compass direction, have a unique shape and do not stack together like conventional coding blocks, as shown on table 5. They have smooth rounded shapes that can be nested within other blocks, fitting into numerical value spaces. The design of these blocks encourages students to explore sensor categories, matching block shapes and colors for incorporation into their programs.

Vignette for Sphero Block-Based Environment

In the vignette below, the students explore different categories of programming blocks, using shapes and colors to support them in navigating the programming environment. In this activity, the students are tasked to modify a Sphero program to determine individual height data for an activity framed around limiting park users from some rides on the amusement park based on their respective heights. This activity was framed such that, for safety reasons, customers would only get on rides they were tall enough to ride. To achieve this, a cut-off height was designed into rides so that riders could be selected only if they were tall enough to ride, (in this case, as many as the students in the class). This required students to measure their

heights as seen in Figure 23. This activity is lesson 6 of the Sphero.Math curriculum, titled "To Ride or Not to Ride," required students to modify a Sphero program to determine individual height data, meaning they will be able to use mode, median, and mean to determine the best height cut-off using their class height data.

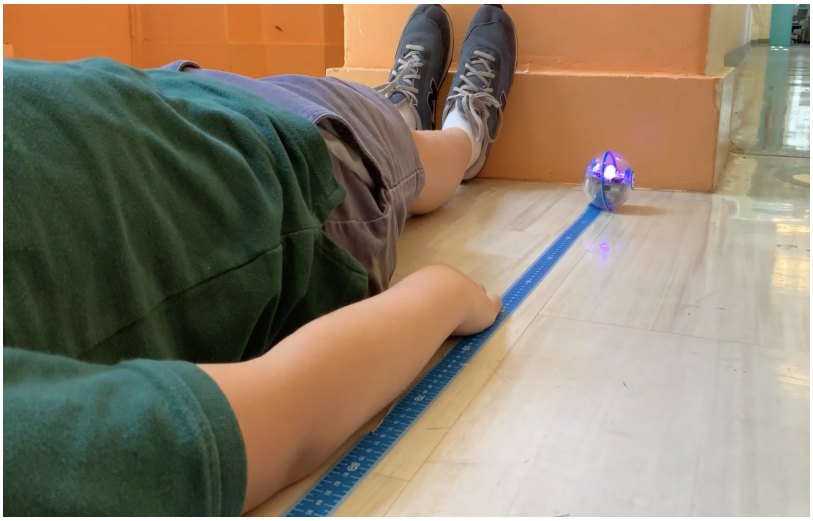


Figure 23: Student measuring his height using the robot and their program.

Students were provided a program (figure 24) to modify and use to determine the heights of each person in the class by finding the length of all the pieces of string, recording lengths, and sharing with the class. To achieve that, the students needed to build a string by constructing a sequence of characters such as letters, numbers, and symbols so it could take symbols or characters (Figure 24). The particularity of this activity is that the Sphero.Math curriculum was designed to engage learners to use the “on collision” event block as seen in Figure 24. When the on-collision block is integrated into a program, it activates the robot’s sensors to detect when the robot

runs into an object (like a wall or any solid object). In this activity, the students were tasked with modifying the program in Figure 24 to make the robot stop and the tablet speaks the distance when a collision occurs. The distance in this case would be the height of the students who would lie on the floor with their feet against the wall to measure heights (figure 23).

To achieve this task, learners need to understand the categories of the blocks, including how to nest blocks into each other.

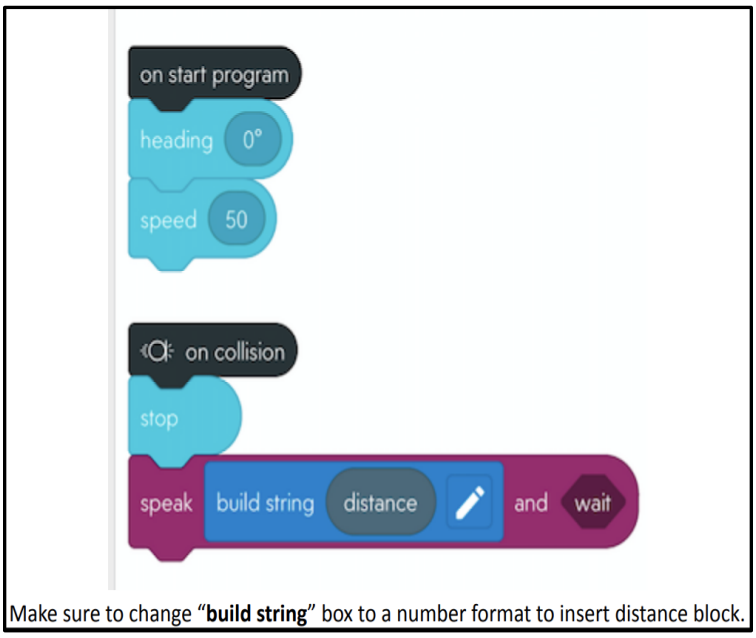


Figure 24: The program provided an activity for students to use to determine the heights of their mates.

The vignette I present below shows a group of three students (all girls) working to modify the program given by the teacher, to determine their height data. Students in this activity had very little experience with the programming environment and tended to rely heavily on the visual cues on the blocks to program.

Vignette on Sphero Block-based Environment/Language

Student 3: Go to sensors go to sensors (scrolling across to find the sensors categories) Let me see.

Researcher: Sensors?

Student 3: Yes, because this is the grey part

Researcher: Look at the shape of that, and then she said there was a pencil here, right?

Student 1: Add string, ok so add number?...

Student 2: No no no no!

Researcher: Wait, What are you going to add?

Student 1: We are going to add distance in a (inaudible)

Student 2: Find the shape of that (pointing to the shape of the space into which they have to fit the nested icon in the block)

Researcher: How will you define the distance?

Student 2: That's the distance, (pointing to the grey distance icon).

Student 1: But it's grey, how are we going to do it? (in a frustrated voice) then dragging out the grey distance icon she asks her friends, "Are you sure? but we need something grey"… continues to search in the blocks icon library and asking "How are we going to do it?"

Student 2: Wait, I think we're supposed to find the shape of distance (then they open the build string blocks and examine the shapes of the blocks in that library)

Student 1: Shape of the distance shape of the distance... (scrolling through different categories of the grey block library) She then says in a frustrated voice "They're just circles."

Student 2: Yeah there was one right there (touching the placeholder for the distance on the screen

Student 1: No no that's not it....this hard

Researcher: So when you put here, look at this (pointing to the build string block)

Student 1: It says add number!!! in an excited voice (screen showing shapes of different input types to add as inner blocks to the build string block)

Researcher: Correct

Sudent 1: Then so we add add that (dragging the pill-shaped block to add numbers into the parent build string block). immediately the shape fits into the block space into which a distance can be added, and the rest of the team screams … "Ohhhh..oh my God, wow, wow, wow " in excitement.

Student 1: Wait so how do you take that part out? (pointing to the distance icon on the parent block).

Student 3: I think you gotta emmm take it away with this, (touching the editable virtual pencil on the block, which gives room to edit away the existing square-shaped block, to only leave room for the oval shape into which they can now fit in the distance command when that happens, they all scream in excitement at the discovery of these new options to find string in numbers and edit the shapes to hold the numbers or variable commands.

Researcher: So where is the block you use to make turns for Sphero?

Student 2: This, the heading (showing a heading block on the screen)

Student 1: It makes Sphero turn right, it can make it turn right (moving the arrows of the virtual protractor on her screen left and then right)

Researcher: So you think if you put the heading in your code…

//Student one then touches her screen with the pencil indicating the position to place the heading block under the speed block.

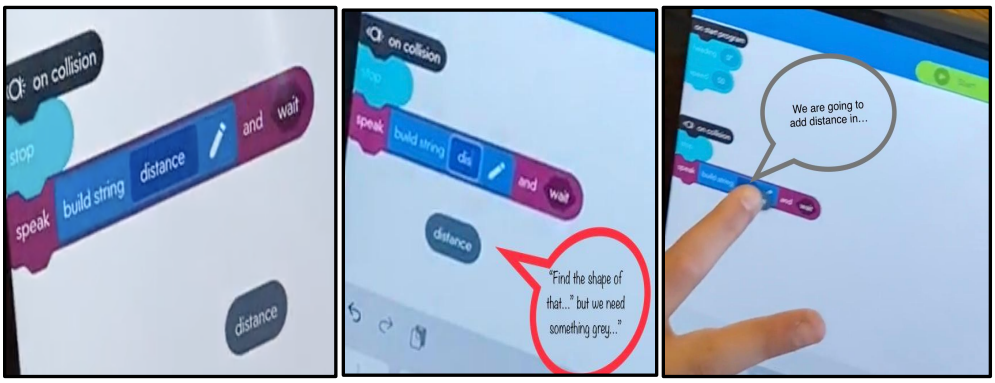
Vignette Analysis on Sphero Block-Based Environment/Language

As mentioned in Chapter 3, learners in this study used Sphero blocks to program the robots. The shape and color of programming phrases offer visual cues as guidance for their assembly (Lin & Weintrop, 2021). These design choices are intentional to make programming more accessible and visually engaging for young learners. Visual features designed into the programmable blocks, help reduce the cognitive load on the students and ‘‘allow students to focus on the logic and structures involved in programming rather than worrying about the mechanics of writing programs’’ (Kelleher & Pausch, 2005, p. 131). These characteristics of visual

programming languages have the potential to enable students to grasp computational concepts easily, eliminating the necessity to master intricate programming syntax. In my analysis below, I share some design choices supported by evidence from vignettes on how the shapes and colors of blocks contributed to scaffolding young learners' CT experiences with the Sphero programming language as follows:

Abstraction: Sphero programming blocks were able to help students understand programming languages through abstraction. I situate abstractions here to Wing's, (2011, p.20) argument that *abstraction empowers computer scientists to manage complexity and scale by capturing essential properties while concealing irrelevant distinctions*. The students in the vignette used block categories to sort out blocks to use. In the Sphero programming environment, types of programming blocks are grouped into categories based on their functionality, such as movement blocks, control blocks, controls, operators, comparators blocks, etc. Student 3 tells her peers during the activity, *"Go to sensors, go to sensors" (scrolling across to find the sensors categories).* When the researcher prompts the students with the question "sensors?", to clarify what they mean by sensors, student 3 responds *"yes, because this is the grey part"* (figure 25). The students were tasked with modifying the program to announce the distance (height) Sphero travels on collision with the wall. In identifying the distance block using the block's color, the learners demonstrate in this vignette that the shape and color of the block can be used to categorize and recognize blocks to use within any category. This way of abstracting and generalizing

blocks using color provides evidence that the choice of categorizing blocks by color provided a visual support for students to navigate through the blocks categories.



(a) (b) (c)

Figure 25: Students use colors and shapes to find and nest the distance block into a build-string block. (a) The pencil icon indicating the block can be edited. (b) Distance icon colored grey (c) Student pointing to the space into which they will fit the distance icon.

To further support the argument that students used the categorization of blocks by color to edit the "build string" block, I argue that the students also categorized by shape and color to help them edit their code. This is seen as student 1 checks with her peer if she had to "*add string,* or *add number?* Student 2 replied, *"no no no no!"* The researcher prompted; *"wait, what are you going to add?"* student 1 replied; *"we are going to add distance in a (inaudible)"*. To further guide her peers to find the "distance" sensor, student 2 says, *"find the shape of that"* pointing to the pill shape of the space into which they must fit the nested icon in the block (figure 25). By organizing blocks into categories with distinct shapes, block-based programming environments were able to provide a visually organized and structured environment where students can memorize and keep track of blocks categories by shape and color. The learners used the CT component of abstraction by *managing* the *complexity* of

the programming environment *and capturing essential properties* (shape and color) of the blocks making it more manageable for them to sort from other blocks and understand their function by color.

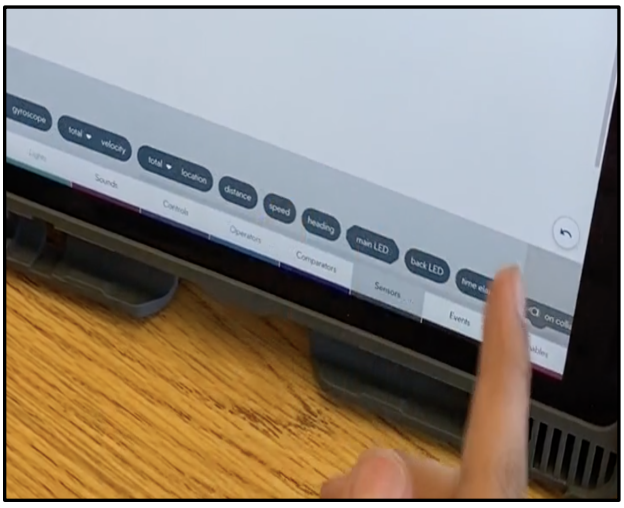


Figure 26: Students swipe on the gray blocks on the sensors categories at the bottom of the screen to find the pill-shaped grey distance block.

Building Algorithms: The consistent use of shapes and colors across various programming blocks supported learners build algorithms (figure 27). Once young learners connected meaning to certain shapes and color combinations for block categories, they easily applied that knowledge to other blocks of the same category to build algorithms. Abstractions using shapes and colors helped learners generalize knowledge about the blocks consequently supporting them to build algorithms. For example, student 1 used color coding to relate to the distance block when she communicated with her mates saying, *"But its gray how are we going to do it?"* (in a frustrated voice) then dragging out the gray distance icon she asks her peers, *"are you sure?... but we need something grey"* … continues to search in the blocks icon library

and asking *"how are we going to do it?"* (figure 26). The learner searches, sorting by shapes and colors as she works on building the algorithm. The students' consistent use of color-coded language to find blocks indicates that the design incorporating color-coding in coding blocks facilitated their ability to abstract block functions using colors and made it easier for them to locate blocks for their code when constructing algorithms.

To further support the idea that colors and shapes supported learners to build algorithms, student 2 said to her peers, *"Wait, I think we're supposed to find the shape of distance"* (then, opened the build string blocks and examined the shapes of the blocks in that library) (Figure 26). Student 1 echoed student 2's words saying, *"Shape of the distance, shape of the distance…"* (scrolling through different categories of the grey block library) she then said in a frustrated voice *"They're just circles".* The use of metaphorical language to describe the distance block using its shape is proof that the use of visually functional shapes familiar to young learners was able to make the programming less intimidating, lowering the floor for young learners to interact with the programming environment. The shapes of the blocks used in this example to build strings helped the students to generalize their knowledge about blocks based on their shapes and colors thus engaging with the blocks and consequently the CT component of algorithm design.

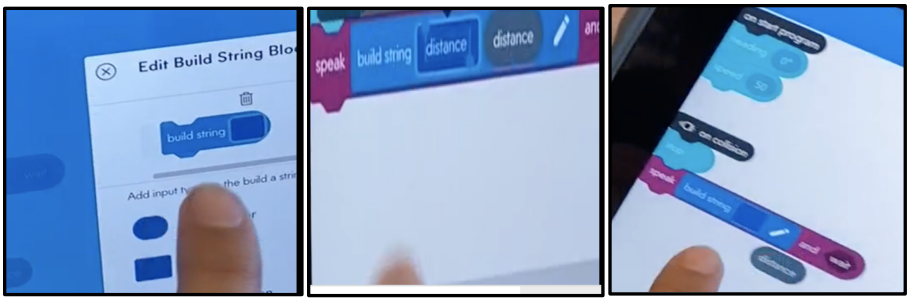


Figure 27: Students opened the build-string blocks to edit their code using shapes and color codes to find the appropriate blocks to nest into other blocks.

The shapes and colors used in block programming environments served as powerful visual aids and abstraction tools that supported young learners' understanding of the different programming blocks, they used to build algorithms as seen in the vignettes. The learners in this example were able to abstract and program complex syntaxes like building strings, and variable types and design programs to help them collect data they can use to calculate the mean, median, and mode for heights in their class. The colors and shapes intentionally designed into Sphero's programming interface supported learners by enhancing categorization, providing visual cues for functionality, creating a generalizable and predictable environment, encouraging feedback, and making programming more accessible to the learners. The abstraction tools centered around the programming blocks helped simplify the programming environment by enabling learners to memorize block categories based on colors and shapes that helped them build algorithms.

As seen from the analysis above, colors and shapes of programming blocks can be used as visual cues to represent the function or purpose of a particular block or category of blocks. For example, the movement block categories are colored blue, while the control block categories are color-coded purple, meaning colors hold

information about the category and function of each block. When learners visualize colors associated with different types of commands, they can quickly understand how to assign and match blocks to different sequences by color and shape to manage the flow of the algorithm.

Conclusion

This chapter sought to answer the question; "*What features of the Sphero robot and its accompanying programming environments facilitate students' understanding of CT knowledge?"*

Here are four design features of the Sphero robot and its accompanying environment that support CT knowledge as analyzed in the presented vignettes; (1) programmable LEDs that provide opportunities for learners to break down tasks, create opportunities to organize and structure components to get visual feedback that helped them recognize patterns. (2) Taillight ("aim") for LED/geocentric orientation, that provided visual cues, such as the "aim" light on the robot, facilitated the involvement of embodied practices, empowering students to establish sensorimotor references. These cues served as aids in relating abstract concepts, like the direction of the robot's movement, to familiar geometric principles. (3) Sphero's virtual protractor, which supported students through the CT component of abstraction to address the geocentric aspects of the Sphero robot. By leveraging certain features designed into the tools, students were able to establish generalizable and meaningful scaffolds through embodied practices, supporting their understanding and accurate representation of angles and directions. (4) The block-based environment/language, that uses shapes

and colors as effective visual aids and abstraction tools, to support learners to construct algorithms. Through shapes and colors, learners were able to abstract complex syntaxes, build strings and variable types, and develop programs. The programming environment offered visual cues for functionality, established based on a generalizable and predictable environment, capable of promoting feedback, accessibility, systematic thinking, and programming for learners (Lye & Koh, 2014; Kafai & Burke, 2013).

The selected vignettes have shown evidence of how various aspects of the Sphero robot's design features contributed to supporting young learners' engagement, problem-solving abilities, and CT skill development as they interacted with the Sphero robot and its programming environment (Resnick et al., 2005; Lye & Koh, 2014). The strategic incorporation of generalizable colors and shapes into Sphero's programming interface has proven to be a valuable tool for supporting learners unfamiliar with programming language syntax to learn to program. I argue that these features designed into the Sphero programming environment make Sphero an age-appropriate programming tool, suitable for young learners (Bers, 2018a; Bers, 2012; Kazakoff & Bers, 2012; Sullivan & Bers, 2015; Bers, 2018a; Horn Crouser & Bers, 2012). The programming environments allowed learners to express skills that included “syntactic, semantic, and schematic knowledge (computational concepts) as well as strategic knowledge (computational practices)” as described in Table 2.3 in Chapter 2 (Lye & Koh, 2014).

Furthermore, the integration of LED lights as visual support tools provides evidence of how technology can be integrated into learning environments to provide

scaffolded feedback to support learners' problem-solving, decomposition, and algorithm development skills using advanced technological tools, making the Sphero robot a ‘‘technology-as-partner in the learning process’’(Jonassen, Howland, Marra, & Crismond, 2008, p. 7). The vignette presented evidence of how LEDs on the robotic device can serve as vital indicators to support students map their thoughts onto programming constructs and real-world concepts. This approach offers a tangible bridge between abstract coding concepts and their practical applications, making connections between programming and real-world simulated activities and potentially assisting students in expanding these dimensions of computational practices to improve their overall problem-solving skills, and possibly support students in extending these computational practices towards enhancing their general problem-solving ability (Lin & Liu, 2012; Ratcliff & Anderson, 2011).

The analysis of the chosen vignettes also highlights the significance of multi-sensory engagement facilitated by the Sphero robot. Students used their bodies to interact with the robot's "aim" taillight. These embodied interactions highlight the robot's design ability for young learners to abstract complex concepts by aligning their bodies to the robot's direction and heading to understand the robot’s movement. These embodied engagements deepened learners’ interactions and experiences with the devices and enhanced opportunities for debugging and refining their precision and accuracy skills. The CT component of abstraction was elucidated throughout these vignettes, providing evidence that the Sphero robot and its programming possess design features that lower the complexity of interactions between the tools and the

learners, making them age-appropriate tools to support young learners to engage with CT.

In essence, the Sphero robot's programming interface intentional design, LED integration, and other multi-sensory engagement potentials of the robot collectively create an environment that supports certain elements of CT integration such as Abstraction, pattern recognition, decomposition, algorithm design, and problem-solving (PRADA).

The most important CT concept that was demonstrated throughout this chapter was the component of abstraction. I argue that specific features, such as the “aim” light, the Sphero robot's design with LEDs visible through its transparent shell, and the categorization of programming blocks using colors and shapes, reduced the complexity of task navigation for learners. Given the significant role that abstractions play in CT practice and skill development, this approach supports efforts to understand how to design activities and tools that can conceal complexity in ways that can advance CT education for learners of all ages (Barr & Stephenson, 2011; Brennan & Resnick, 2012; Grover & Pea, 2013). Through its design and carefully crafted features, the Sphero robot makes an effective educational tool that empowers young learners to abstract, grasp programming concepts, connect them to real-world scenarios, and interact with different dimensions of CT concepts and practices. In the next chapter, I analyze how the design of the Sphero robot and its programming environment supported learners’ interactions through forms of representations learners used to navigate the learning environment.

Chapter 5: Findings

In this chapter, I present findings from selected vignettes on different forms of representations learners interacted with to ground their understanding of CT knowledge. Focal students selected for these observations used several forms of representations to understand, navigate, and solve CT problems. Given the age group of the learners, they used creative ways to interact with the CT environment due to the level of cognitive complexity required to engage in the tasks. I observed that students engaged in translating between various forms of representations. Depending on the complexity of the tasks, students exhibited different levels of maturity in their ability to represent concepts.

To analyze my second research question, which seeks to understand; *What kinds of representations and expressions do young learners engage to interact with Sphero robots in a mathematics classroom and how do these representations contribute to successful problem-solving in CT tasks?* I analyze different moments when the students expressed different forms of representational knowledge during the activity either through verbal expressions, collaborations with peers, written out artifacts, information entered on their iPads in the form of programming or their notebooks, and even gestures they made as they interacted with the activity. I did not control any variables in those events and settings making this kind of data particularly attractive for analysis using certain coarse grain analysis aspects of IA perspectives.

I analyze the data using the Lesh Transition Model (Lesh and Doerr, 2003) to support findings around the use of representations and representational fluency.

Five groups of students were observed and recorded as they used representations while interacting with Sphero robots. They used these representations to discuss their understanding of tasks or mathematical concepts, navigate elements of CT, program the robots, and share their perceptions of the robots' behavior. The types of representations used varied among the different groups. Students used multiple representations in the form of concrete, graphic, language, symbolic, and embodied expressions to translate across 2D and 3D representations in the activities. In certain cases, students used models and modeling to frame ideas and concepts using real-world models, in ways that helped them understand the tasks.

In the vignettes that follow, I explore different forms of representations learners engaged in as they worked on activities in the Sphero.Math curriculum, inspired by amusement park experiences that guided students to manipulate the robot. This involved programming the robot's movements and measuring the distances it covered. I examine how these interactions enabled students to engage with CT practices and concepts within the context of the activity space.

In addition to providing evidence of how representations and expressions support the understanding of CT knowledge, I will be emphasizing the idea that representations, especially in learning contexts, are rarely standalone entities. Instead, they are part of larger, interconnected systems that are either unique to an individual (personal and idiosyncratic) or shared by a group or culture (cultural and conventional) (Goldin, 1987; Lesh, Landau, & Hamilton, 1983). To answer my research question one, which seeks to understand "*what kinds of representations and expressions do young learners engage to interact with Sphero robots in a*

mathematics classroom and how do these representations contribute to successful problem-solving in CT tasks? I use the following six forms of representations described below to analyze how students in robotics activities explored CT in mathematics contexts. These five forms of representations include:

1. Concrete manipulatives: Concrete manipulatives that allow students to interact with and explore CT and mathematical concepts.The Sphero robot is an example of a concrete manipulative.
2. Graphical Representations: Diagrams or maps that help students visualize spatial relationships or problem-solving strategies.
3. Language Representations: Language representations are representations that support understanding, communication, and reflection on CT and mathematical concepts.
4. Symbolic representations are abstract, non-concrete representations that use symbols, signs, or notation systems to convey meaning or express relationships between concepts.
5. Embodied Representations: Embodied representations are learning tools or experiences that involve physical engagement and interaction with the environment or objects, allowing learners to connect abstract concepts to their bodily experiences.

In my findings, I do not present each of these representations as if they are experienced in isolation. I try to demonstrate using the data that emerged, that representations are a part of a larger system that helps students develop a deep

understanding of computational thinking and mathematical concepts. In this chapter, I will present 5 vignettes to highlight how various representations are used to support CT acquisition. Specifically, across these vignettes, I am going to present each of the five representations discussed above, each of which will have one vignette. Below is a list of artifacts and experiences that will be tied to each representation. I provide a prototypical example to provide a context when learners are programming the robot to navigate a maze. In this context, students will typically engage in one or more of the following representations presented in the table below.

Table 6. Five forms of representations and their contextual uses

Representation	Description	Example
Concrete Representation	A visual representation of the activity or artifacts that support spatial reasoning and problem-solving even without programming the activity.	Activity map on a large sheet of paper that provides a visual representation of the activity. Physical robot which, already by design, offers a concrete representation of an artifact.
Verbal Representation	Verbal explanations between group/team members facilitate communication, and collaboration, and share	Language expression representations are used to describe, express, and explain create algorithms, and control the robot.

	ideas and perceptions about concepts and practices	
Graphical Representations	are Diagrams or maps that help students visualize spatial relationships or problem-solving strategies.	2D Grid based map serving as the maze.
Symbolic Representation	Symbolic representations refer to the use of symbols to represent or convey information, concepts, or entities. Symbols are abstract, arbitrary representations that carry meaning within a particular context or system.	Dots, Hashtags, taillight on the robots, and other symbols used to convey meaning during the activity. symbols such as "+" (addition), "-" (subtraction), "=" (equality), and "%" (percentage) are used to express mathematical operations and relationships.

Embodied Representations	Using parts of the body to engage with the activity.	Embodied experiences with the robot and the learning environment. For example, using the arm to gesture directions of the robot, and pointing gestures to focus the attention of peers during an activity.

My findings show that representations occurred in a structured highly interconnected system. I observed that representations unfolded in an organized manner, aligning with specific contexts to facilitate the construction of meaning during interactions as they naturally evolved within the activities. The interconnectedness of these representational systems in CT environments work together to support the development of CT skills by complementing and reinforcing each other. I use evidence from vignettes to demonstrate ways in which interconnected representational systems can support the development of CT knowledge. Considering the multimodal nature of the activities, the richness of the data, and the varied perspectives offered by learners due to the in-situ nature of events, I have organized the findings to leverage the potential benefits that representations can offer in CT knowledge acquisition. Below, I present a series of

vignettes that illuminate how learners use various forms of representations in selected episodes of transcribed video clips. I provide contexts within which these representational structures fall in terms of supporting learners in the Sphero.Math curriculum (Appendix A) and the alignment of these tools with the DCPS common core standards for mathematics.

For each form of representation, I start by providing a description of the representation, providing context for its use with respect to the context of the activity (framing). I follow up with a vignette that provides evidence of how students use representations to support their thinking as they interact with the activities and program the Sphero robot. I provide an analysis of the vignettes providing, highlights of instances when representations supported the development of CT skills and mathematical concepts. Furthermore, I explore the interconnectedness of representations as they unfolded within the vignettes. I conclude with a brief discussion of the role representations played in supporting learners to accomplish tasks.

Use of Concrete Representations – Manipulatives and Objects

Description of Concrete Representations

Manipulatives can make effective learning tools in CT environments. Hands-on learning with tangible objects can be traced back to the theories of Piaget (1970), Bruner (1966), and Montessori (1917). Tangible tools have proven to be beneficial in facilitating conceptual understanding, promoting engagement, and supporting the development of problem-solving skills in young learners (Bers et al., 2002; Ching et

al., 2018; Sullivan & Heffernan, 2016). I argue in support of Uttal et al., (1997) who suggests that when children interact with manipulatives, they interpret them as representations of the abstract concepts they are learning, which helps them make sense of these ideas in a more tangible and accessible way.

In this study, I position the use of manipulatives as a form of representation in CT tasks because they function as iconic representations, providing perceptual motor grounding and offering external scaffolding to support learning.

Learners can interpret tools as tangible representations of abstract concepts which helps them develop a deeper understanding of underlying principles that may require the application of CT practices to support problem solving. The tasks and activities designed in the Sphero.Math curriculum involves the use of concrete objects within the physical space of the activity. These manipulatives and objects are arrayed to correspond to spatial configurations in ways that the children's representations of these objects can create relationships between the objects and the activities. This use of manipulatives aligns with the DCPS mathematics common core standards SMP4 - Modelling with mathematics, which prescribes "the use of concrete objects and visual models to add and subtract fractions".

Concreteness of Sphero.Math Materials and Description of the Activity

The Sphero.Math curriculum involves hands-on concrete materials for students to interact with. Activities will typically require students to use a Sphero robot described in Chapter 3, (Figure 3.2), the Sphero.edu app (Figure 3.2), and additional materials (e.g., masking tape, maze map, meter sticks, rulers, protractors).

Students use these materials to solve tasks modeled to resemble real-world problems. Concrete representations in this study are activities designed on large sheets of paper or using other arts and crafts materials to model real-world situations as described in the Sphero.Math curriculum. The Sphero robot's friendly spherical shape and the colorful programming interface where learners primarily use Sphero's roll block (Figure 3.3) to control Sphero's movements make a compelling concrete representation of a manipulative to introduce into CT learning environments with young learners.

However, for this study, I positioned the art and craft materials such as the large activity sheet as the concrete manipulative with which the students used the Sphero robot to interact. Despite the tangibility of the Sphero robot and the centrality of its role within the activities, the robot is used mostly in metaphorical terms. For example, the robot is introduced into the activities to metaphorically represent a moving object such as a car, a train, a human, and other forms of industrial robots interacting with objects designed for the activity. The robot analogically represents mobility within the activity, while the activity sheets provide opportunities for learners to physically manipulate and interact with the activity by measuring distances, counting squares, and visualizing the robot's actions before making decisions to enter data into the Sphero.edu app for the robot to execute. I therefore focus on the grid map representing the monorail stops as the concrete manipulative to serve as concrete representation in this study (figure 5.1).

Vignettes with Students Using Concrete Representations

I present a vignette showing how concrete representations such as hands-on artifacts and concrete physical tools were used during the activities. The framing of the activities around amusement park themes engaged the students to use a grid map designed by their teachers, yardsticks, and any other learning materials they found useful in the classroom to help them measure distances and make conversions into centimeters given that the Sphero.edu app exclusively measures distances in centimeters.

Activity: Monorail stops. The Monorail stops activity is lesson #8 on the Sphero.Math curriculum. This activity is framed around the idea that visitors who visit a park would need a monorail train to move around the park especially if it is a big park. For this activity, students are tasked with using the Sphero robot to program a monorail's path and advance to use loops when they recognize repeated patterns on the path. Students begin by programming shorter paths and the teacher incrementally extends the length of the paths and number of stops as the students grasp how to recognize patterns and use loops.

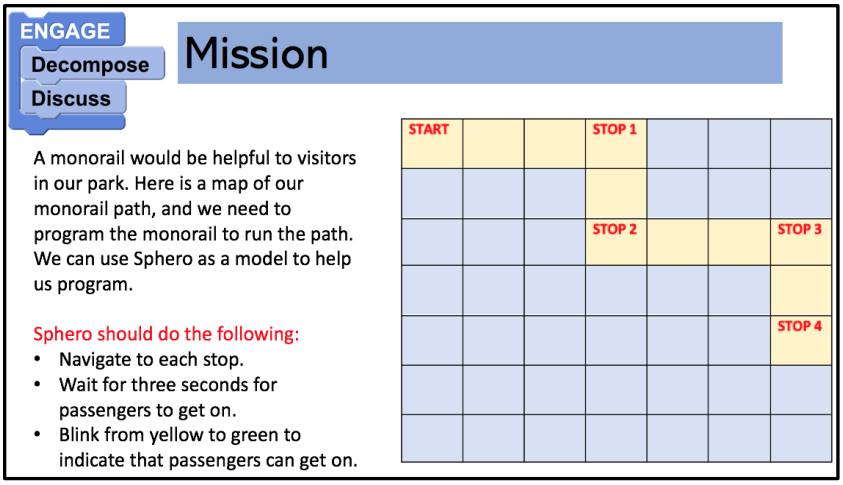


Figure 28: Sphero.Math monorail activity sheet.

A map of the monorail is provided in the activity (Figure 28) with instructions on what the Sphero should do. Students are tasked with coding the Sphero robot to navigate a fixed path by manipulating the angle speed and time in the Sphero roll block. This activity is designed to help students understand the relationship between turning in different directions and angles, find patterns, and use loops to program repeating patterns.

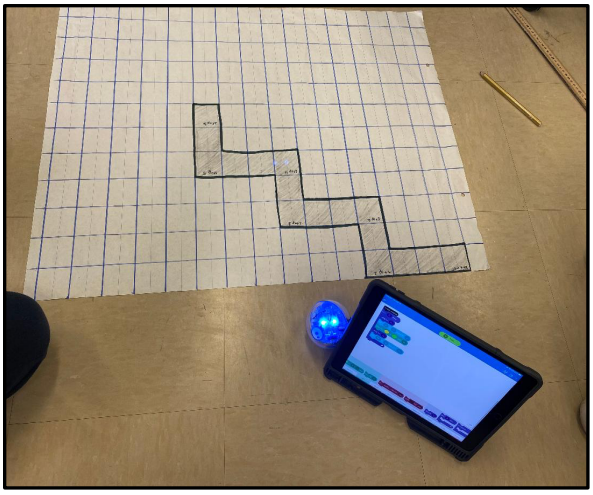


Figure 29: Monorail activity with a robot and iPad provided by the teacher at the start of the activity.

The vignette script on concrete representations presents the monorail as the concrete manipulative. Three students work together on the activity board, to program the robot to navigate the monorail path. In this vignette, the students interact with the grid map as a concrete manipulative (figure 29) provided by the teacher in an activity to mimic monorail stops as shown on the activity sheet (Figure 29). The vignette captured between the 1st and 3rd minutes into the activity presents three students: two boys and a girl using concrete manipulatives to complete tasks.

Vignette Script for Concrete Representations

Researcher: So, what are you trying to do?

Student 1: So, we are trying to make Sphero do this (touching the maze with her pencil) so we're going to make it roll we're trying to figure out the speed and how much heading we need to go.

Researcher: how much heading?

Student 1: heading is like turning

Researcher: how much turning?

Student 1: yeah

Student 2: like you are in a car and then you turn right.

Researcher: you are in a car and then you turn right?

Student 2: Yeah, or the other way

Researcher: and how do you know when to turn or, how do you know when to tell Sphero to turn?

Student 1: erm, we program it and we put the heading blocks as much times as we need it,

Researcher: so where is the block you use to make turns for Sphero?

Student 2: this, the heading (showing a heading block on the screen)

Student 1: it makes Sphero turn right, it can make it turn right (moving the arrows of the virtual protractor on her screen left and then right)

Researcher: so you think if you put the heading in your code..

//Researcher observation: student one then touches her screen with the pencil indicating the position to place the heading block under the speed block

Student 1: lets make the speed be.. what do you think the speed could be?

student 3: I think it could be in the middle like not too slow and not too fast

Student 1: our recent speeds are like 15, 2 emm 3, 15 i mean... which should we do 15? then I put the heading, (pulling the heading block unto the block canvas) saying, that will go down

//Researcher observation: And then the virtual protractor pops up where she drags the arrow to the downward position (180 degrees).

Student 1: we also needed to stop over here first, (pointing at the screen on a position within the code)... oh we forgot roll, where's roll, and the pulls the roll block from the movements category

//Researcher observation: Using her notebook, she looks at the steps on the activity, saying " roll for ...

Student 1: That's how many seconds we want it to be? roll for? how many seconds? 16s.

Researcher observation: Then looking at the activity, sheet, she points with her pencil saying

Student 1: "So we want it to go that way .. no not for 16s! Sixteen seconds like for 5s will be good I guess

Student 2: once it goes that way, we want it to go down

Student 1: //writes in notebook saying "roll, at 14s for

Student 1: Now we now want it to repeat that right?

Student 3: Yeah, because if we didn't repeat it it will be kind of boring if we just stopped like that. Now we just have to do this loop, one two three four

Student 1: Yeah, four more times

Student 3: I have seen four but I am still thinking five

student 2: You wanna go five times?

Student 3: not sure

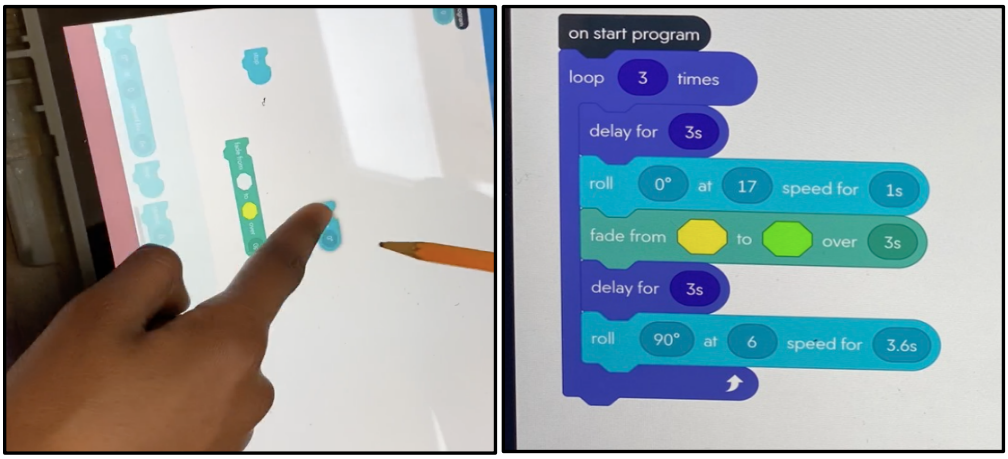
Student 2: It will be more fun

Researcher: So, five is as a function of what?

Student 1: five is as a function of how many times it will do this coding in blue (touching the codes nested into the loop blocks) (figure 30)

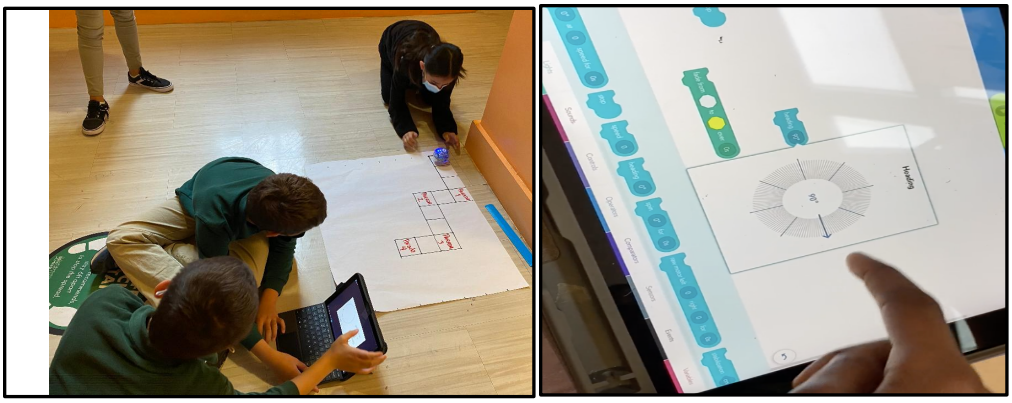
Researcher: So, where's that on your board?

student 3: Oh that's six, that's six stops, one two three four five six



(a) (b)

Figure 30: Students drag and drop codes onto the coding canvas. (b) Codes nested into the loop block.



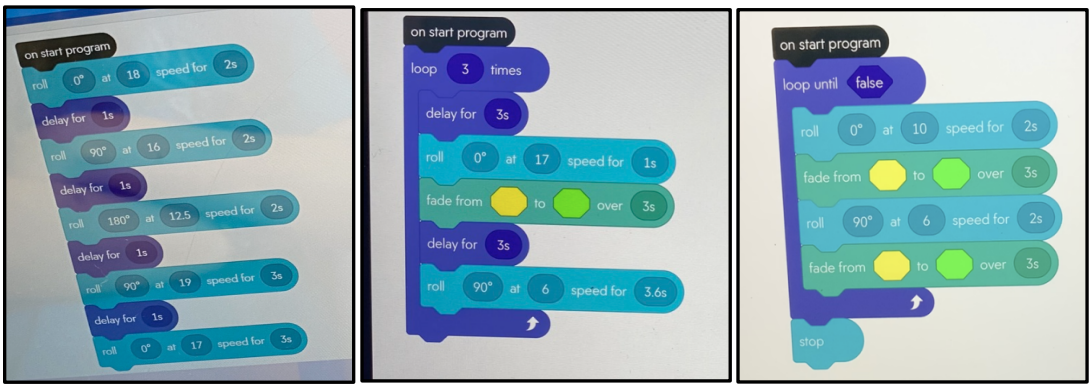
(a) (b)

Figure 31: (a) Students engage in the grid map as a concrete representation. (b) Students move the arrows of the virtual protractor to find a matching angle with the turns on the map.

In this vignette, the use of concrete representations specifically the monorail activity path (Figure 31a) proves helpful in providing visualizations that create pathways for learners to interact with both physical and virtual tools such as the robot and the virtual protractor (Figure 31b). This aspect is significant in supporting the students' understanding of CT concepts. The design of the monorail path with distances and stops that could be incrementally added over time supported the students in decomposing the activity into parts, recognizing patterns introduced, and using loops in their programs.

The concrete nature of the monorail path supported the students to use this form of representation to engage in two CT dimensions. First, the students engaged in CT practices such as decomposition, breaking down and programming the activity in parts then debugging and refining their ideas among themselves. Secondly, the students engaged in a concept of pattern recognition and looping when they identified a repeating pattern on the path. All these practices can be seen in Figures 32(a) and 32(b) and 32(c) which show how the students refined their code, transitioned from

writing single lines of long code to using loops. Thirdly, the concrete representation provides an opportunity for the students to have conversations about their perspectives by expressing ideas and asking questions about the activity.



(a) (b) (c)

Figure 32: (a) Single lines of code written by students to navigate the monorail path. (b), (c) Students after recognizing a pattern used loops to nest blocks.

Initially, at the start of the activity, student 1 articulates their intention for the Sphero's actions by referencing the monorail path, establishing a connection to the concrete representation. Student 1 says, *"So we are trying to make Sphero do this* (touching the maze with her pencil) *so we're going to make it roll we're trying to figure out the speed and how much heading we need to go"* When the student traces the path with her pencil, she establishes a tangible link with the maze, a concrete representation of the task. This interaction with the physical manipulative reinforces the student's connection with the problem and offers a visual guide for planning the Sphero's movements. This connection with a concrete visual reference (the monorail

map) provided opportunities for learners to share their perspectives by expressing ideas and asking questions about their CT engagement or practice.

The researcher further prompted the student, referring to her explanations around the monorail path by asking *"How much heading we need to go"*, the student used a metaphor to express her thoughts saying, *"Heading is like turning"*. The researcher further prompted: *"how much turning?"*. In a series of questions and answers that ensued, the researcher tried to understand what the students meant by *"turning* "on the path. Again student 2 brought in another metaphor saying, *"like you are in a car and then you turn right or the other way"*. At this point, the students are organizing and verbalizing their perspectives on the series of steps and actions the robot would make to complete the maze. Student 2 likened the actions of the robot to the actions of a car.

The researcher further inquired from the students to understand how they would determine when to turn; *"how do you know when to tell Sphero to turn?"* still referring to the directions they described on the path. To that, student 1 said; *"erm, we program it and we put the heading blocks as much times as we need it, it makes Sphero turn right, it can make it turn right* (moving the arrows of the virtual protractor (symbolic representation) on her screen left and then right. Student 2 used the virtual protractor to explain how the Sphero robot will turn, providing evidence of how their perspectives and conversations around the monorail path supported their interactions with virtual and physical tools within the activity and consequently scaffolding their understanding of an angle.

Furthermore, the learners by visualizing the monorail path, were able to recognize patterns on the path, as student 3 says *"yeah, because if we didn't repeat it will be kind of boring if we just stopped like that; Now we just have to do this loop, one two three four"*. Student 1 continues by saying *"yeah, four more times"*; Student 3 then continues saying *"I have seen four, but I am still thinking five".* The students were able to find a pattern and decide to use loops. The vignette shows student 2 asking the question *"You wanna go five times?* And student 3 answers "*Not sure"*. This means the students are anticipating the possibility of uncertainty and possible errors, meaning they think about debugging as they code. The monorail path offered a tangible depiction of an angle through directional cues visible on the track. This encouraged students to link their interpretations on the map with abstract maneuvers on the virtual protractor. By aligning the directions on the physical monorail path with the virtual protractor, the learners effectively identified the accurate angle values to input into their code.

This vignette presents proof of how young learners rely on concrete representations to understand complex concepts like angles, engage in reasoning about CT concepts and practices, and offer perspectives on their interactions with the activity. Concrete representations were able to present the learners with physical, tangible objects they were able to rely on to connect to abstract symbols such as the virtual protractor. This representational translation afforded by the concrete manipulative was able to support learners to engage in CT components of decomposition, and pattern recognition leading them to use loops. The concrete monorail path provided the required support the students needed to translate

information from the physical space to create algorithms that allowed them to develop accurate predictions of how the robot would behave and anticipate potential errors or problems in their code. This is evidence that the students used concrete representations as scaffolds to help them develop their code.

Interconnected Representations in Concrete Representations Vignette

In this vignette, I noticed the use of concrete manipulatives supported the use of other forms of representations such as symbolic representations, embodied gestures, and language expressions. An example of this interconnectedness is seen in the vignette when the students discussed how to program the Sphero, using the abstract concept of "heading" to describe the direction the robot should move. From the vignette, student 1 discussed with teammates a strategy that leads her to connect between language expressions of the abstract concept of heading, which she breaks down using a visual image of a protractor to visualize the angle. From the vignettes, the student says "*Our recent speeds are like 15, 2 emm 3, 15 I mean... which should we do 15? Then I put heading"*, *(pulling the heading block unto the block canvas) saying, "That will go down"//and then the virtual protractor pops up where she drags the arrow to the downward position*. The student at this point in this vignette is grounding her understanding of the "heading", which determines the direction (angle) the robot will travel using language, and then pulling out the symbolic image of a protractor to make pointing gestures saying, *"we also needed to stop over here first"*, *(pointing at the screen on a position within the code)*.

In my analysis of this vignette, I argue that using interconnected forms of representations (symbolic representations, embodied gestures, and verbal

expressions) the students translated across different forms of representations ground their understanding of the concept of heading. The virtual protractor and the heading blocks served as symbolic representations that were abstracted and interconnected using other forms of representations to support the learner's understanding of headings, angles, and turns; aspects that were crucial in helping them build codes to control the robot's movements.

Discussion

The use of concreteness of the representations provided by the monorail path in this vignette served as an important scaffold for the students to develop their CT skills and problem-solving skills. The concrete representations served either as visual guides or scaffolds to support and provide feedback to the students about the accuracy of their programs. By engaging with these concrete manipulatives, students were able to decompose the tasks into smaller manageable steps, organize which parts come first, find patterns, and overcome abstractions in ways that helped them develop CT components of pattern recognition, decomposition, and abstraction. This vignette shows students thinking about debugging or possibilities of error as they problem-solve. Additionally, the students were able to lean on the visual feedback provided by the monorail path to engage in conversations and interactions with the tools provided by the activity. Learners shared their knowledge and ideas about the task thereby providing their perspectives about the activity and the technology they interacted with. Computational perspectives are an important aspect of CT dimensions that can support students' understanding of their thoughts and the technology they are interacting with, as described in Table 4 in Chapter 3.

Overall, the use of concrete representations, connected with other representations served as an important cognitive support tool for the students to complete tasks, and engage in several dimensions of CT practice such as abstraction, pattern recognition, decomposition, and algorithm design.

Graphic Representations

Description of Graphic Representations

Graphic representation refers to the use of visual elements, such as diagrams, charts, maps, or graphs, to display information. These representations allow for understanding of complex data or concepts by providing a visual summary or illustration of ideas, and concepts of facts (Lakeoff, 1987). Graphic representations are often used to make information more accessible and easier to understand, especially when dealing with large data sets or intricate relationships between variables (e.g., Lakeoff, 1987; Cuoco & Curcio, 2001; Romberg, Fennema, & Carpenter, 1993; Yerushalmy & Shternberg, 2001; Zimmermann & Cunningham, 1991).

I position the monorail path as a form of graphic representation since it has visual elements and diagrams that take the shape of a maze the robot must navigate (figure 33). Graphic representations can help visualize complex relationships or patterns. In this study, I suggest that graphic representations in robotics-mediated CT tasks can be understood through the following aspects:

Encapsulation of concepts: Graphic representations can encapsulate complex CT and mathematical concepts within a concise and standardized notation system. For example, a single square on the maze in a robotics activity can store data related to variables such as speed, time, and the total distance the robot must travel between stops. This encapsulation of information, once understood, can facilitate efficient programming using loops, communication among peers, reasoning through the problem, problem-solving, and reducing the complexity of the task thereby enabling learners to focus on other tasks within the activity.

Generalization and transfer of knowledge: Graphic representations can support the development of certain CT concepts such as decomposition, pattern recognition algorithm design, and consequently the translation of different concepts between contexts. Learners can decompose and program parts of the maze, and once they establish a pattern, they can use loops to program. By abstracting away from specific concrete instances or scenarios, graphic representations provide visual imagery that can allow learners to recognize patterns, principles, and relationships that are applicable across various situations.

Connection to real-world applications: Graphic representations can serve as the bridge between abstract CT knowledge and real-world applications of CT. By translating abstract concepts into a graphic form, learners can apply their understanding of these translations to solve problems, create algorithms, or develop computer programs.

Graphic Representations Activity

An example of a graphic representation in the context of programming the Sphero robot is the Monorail Stops activity, as described above. Students are tasked with programming the Sphero robot to navigate a fixed path by manipulating the angle speed and time on the Sphero roll block. The maze used for this activity is a visual graphical representation that uses symbols, squares, colors, and lines to guide the flow of control and the sequence of operations that the robot is programmed to follow. The students were tasked to design the instructions to follow sequences of operations represented on the maze in the form of an algorithm. Specific symbols in the activity represent different types of operations or decisions, such as

1. Colored squares: Represent the path.
2. Stops: Instances where the Sphero must stop (i.e., delay blocks)

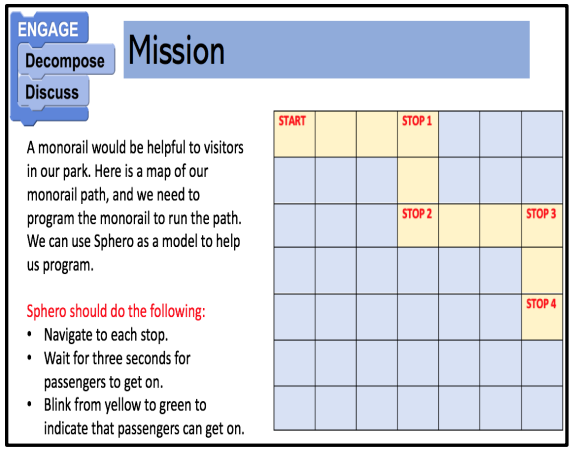


Figure 33: Grid map for monorail stops indicating colored squares and stops that serve as symbolic representations.

The maze serves as a graphical representation of the algorithm, helping students visualize the relationship between variables, organize steps, decompose tasks, find,

and visualize patterns, and develop a flow of sequences of actions to be performed by the robot.

As learners engage with the maze, they can convert visual data from the physical space into the virtual space. This process involves interpreting details like distances between squares on the maze and converting them into speed and time within the visual programming language on the Sphero.Edu app to control the robot. The robot's programming interface has visual programming symbols that support the connection between the robot's actions and the graphical representations on the maze; these include symbols or blocks that represent various commands, such as:

1. Move forward.
2. Turn right.
3. Turn left.
4. Loop (repeat a set of commands a specific number of times)
5. Conditional (execute a command if a certain condition is met)

Based on learners' understanding of these commands, they can create a sequence of steps or blocks to form an algorithm that guides the robot from a start position on the maze to a stop position. For instance, students can use the roll block (Figure 4) to navigate the robot forward.

By interacting with the graphic representation, supported by a programming language, students can develop important CT skills, such as decomposition, pattern recognition, and algorithm design, while directly controlling the robot's movements.

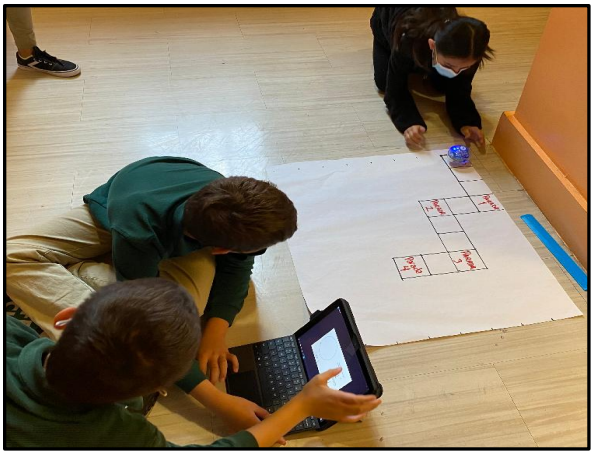


Figure 34: Students working on a maze with symbolic representations of Start and Stop indicated on the map.

Vignette Foregrounding Graphic Representations

The vignette below illustrates how three students, two boys, and a girl, use a maze provided by their teacher to complete the monorail stop activity. This vignette was cut three minutes into the activity as the students worked on programming the Sphero to navigate the maze. The students start by talking among themselves to plan the task, breaking down the steps they may need to accomplish the task.

Vignette Script for Graphic Representations

Student 1: I think we need speed and time she (the teacher) said that and probably the distance and we need to find when to stop we also need to blink two times

Student 2: Blink from yellow to green

student 1. This is what we need to do okay;

Student 2: Yellow, green (while writing in her book)

Student 2: We need roll, heading, speed,

Student 1: We need roll block right?

Student 2: Yes,

Student 1: Roll block(while writing in his notebook)...We need roll block what?

Student 2: Roll block, speed and time, heading stop, turn yellow to green

Student 1: Don't we need? We need speed first;

student 2: Yeah,

Student 1: Speed and then heading then speed.

Student 2: Three stop, stop for three seconds

Student 1: Heading... I just put heading.

Student 2: We need to find how to blink, how to...(then reading the writing off her book) speed and time, stop, blink yellow to green stop for...(student 1 interjects)

Student 1: It just needs to be then put time

Student 2: Oh yeah, its first, heading speed and time then erm then stop for three seconds and then blink yellow to green...then she moves over to the activity and moves her finger over the path on the maze

Student 1: There's no time on it? help her find the block of time, he can't find it. speed time, blink yellow-green.

Student 3: (//who had been scrolling on the tablet on the Sphero app to find the different blocks); This is the first one right?

//Student two then counts the squares on the activity sheet to get a sense of the distance to enter into the roll block, and counts up to seventeen. "two, four, six, eight, ten, twelve, fourteen, sixteen, seventeen."

Student 1: The speed will not be seventeen

Student 2: let's see

Student 1: No. you need to stop, this is how you do it for 17 seconds

Student 3: No two seconds, 2 seconds

Student 1: It's 3 seconds, see, one two three (counting the squares on the maze path// assuming each square equals 1second) ... then four five, then six seven eight nine, then ten eleven

Student 2: So three is more efficient

Student 1: Yah or four, probably 3 is more efficient

Student 3: Let do blink

Student 1: yah lets do blink first

Student 3: Yellow light?

Student 1: First is yellow then blink

Student 3: For three seconds? three right?

Student 1: Three seconds

Student 2: Do stop and wait for three seconds

Student 3: Stop and what?

//Researcher observation: student 3, working on the tablet, entering the blocks and the values into the blocks as his teammates suggest.

Student 2: Stop and wait for three seconds

Student 1: Stop and then wait for three seconds

Student 1. Maybe we should stop and wait for 10 seconds instead so the people have enough time to get on

Researcher: So how do you determine the number of seconds you want to stop or start?

Student 1: because we counted the blocks

Researcher: Where did you count the blocks?

Student 1: Here (moving his fingers on the blocks on the activity sheet and counting, one, two...)

Actually 1,2,3,4 that's what where we think it would stop.

In this transcript, students are engaged in a robotics and CT activity that involves navigating a maze that mimics a monorail with stops. The students start by having conversations to determine the actions needed to accomplish the task. These conversations involved details about the path and actions such as counting the number of squares on the path; Student 2 counts the squares on the maze, immediately converting the distance to time. *"two, four, six, eight, ten, twelve, fourteen, sixteen, seventeen"* student 1 then says, "*the speed will not be seventeen"....you need to stop, this is how you do it for 17 seconds".* At this point, the counting of the squares

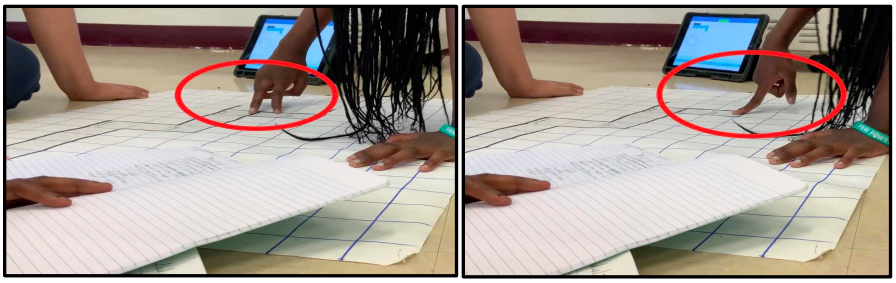
quickly translates into a conversation about time as student 3 says; *"no two seconds, 2 seconds"* student 1 then says; *"its 3 seconds, see, one two three"* returning from seconds to counting the squares on the maze. Here the students are using the graphic representation of the maze to abstract the concept of distance which is defined in the Sphero roll block by distance and time.

Learners attribute a value to each square to be equal to 1-second meaning that they have made some mathematical calculations between the distance of each square to be equal to one second "... *then four five, then six seven eight nine, then ten eleven"*. When the students associate each square on the maze path with a unit of time (one second), they count the squares on the path to determine the appropriate time values for stopping and speed adjustments. This vignette showed how graphic representations supported learners to encapsulate mathematics and CT concepts in mutually supportive ways. The students use addition and iterative logic to mathematically express and enact CT practices of pattern recognition by counting the squares, converting the distances into time, and algorithm design by expressing data into a form that a computer can execute.

Additionally, student 3 interacts with the tablet and the Sphero app, searching for blocks needed to program the robot. Student 2 suggests to Student 3; *"do stop and wait for three seconds".* Student 3 then asks, "*Stop and what?" //works on the tablet, entering the blocks and the values into the blocks as his teammates suggest*. The graphic representations on the maze provided a visual tangible opportunity for the students to plan and execute the task using blink lights to represent delay and stop instances that were other commands to input into their program to indicate when

passengers would get on and off the train. The stop points on the maze helped students to break down the task into steps. Furthermore, students used the graphical representation of the Monorail path to visualize simulate, and engage in mathematical practices when student 1 said; *"So how do you determine the number of seconds you want to stop or start?"* And then student 1 answered*; "because we counted the blocks."* This association between time and counting the squares is further reinforced when the researcher asks the question; *"So how do you determine the number of seconds you want to stop or start?"* And then student 1 answers*; "because we counted the blocks."*

Within this transcript, students actively interact with the maze, exploring variables, leading them to rely on various mathematics and CT practices. The students provide evidence of their interactions with CT practices such as sequencing, organizing steps of the activity, iterative counting, and the association of each square with a time value to mutually support Mathematics and CT practice. Overall, the graphical representation of the activity using a maze supported the students to decompose tasks into parts, recognize patterns, and write algorithms.



(a) (b)

Figure 35: Students write down actions the robot will have to make in a notebook (a). Student moves her fingers over the maze, mapping and embodying the path (b).

Interconnectedness of Graphic Representations with Other Forms of Representations

During their interactions with graphical representations, learners engaged with other forms of representations to support their interactions with the maze. For example, student 2 used symbolic representations by writing in her book (figure 35a). She writes down terms and draws symbols for *"roll block," "speed and time," "heading," and "stop."* The students discussed the order of actions needed to accomplish the task, such as determining the speed first, heading, and stopping for three seconds at each station. Learners organize an execution plan in a step-by-step sequence to ensure proper navigation through the grid map, thus decomposing the problem into a series of sequential steps using language representations.

The students use language representations to verbalize their thoughts about the blocks they need in a way that describes the program flow and sequence of events. Student 2 says *"Roll block, speed and time, heading stop, turn yellow to green"*. Student 1 says, *"Don't we need? we need speed first"*; Here the students are decomposing the task, determining which part comes first. The students discuss programming lights to blink from yellow to green, *"stop, blink yellow to green, stop for…"* indicating that they are using LED colors also as a form of symbolic representation for signaling an action or communicating different actions during the task.

Furthermore, the graphic representation of the maze provided opportunities for learners to engage with embodied representations. For example, when the researcher asked the student, "*Where did you count the blocks?"* student 1 clarified

using embodied representations to support her thinking saying, " *Here". (moving her fingers on the blocks on the activity sheet and counting, one, two...)* and then saying, *"Actually 1,2,3,4 that's what where we think it would stop"* (Figure 35b). This iterative refinement of the students' thought process through their interactions with the activity maze, shows that the graphical representation of the maze supported the emergence of other forms of representations that helped students abstract to build algorithms.

Additionally, the graphic representations supported students to engage in embodied representations using fingers to trace the path on the maze, mapping the path with her fingers; *then she moved over to the activity and moved her finger over the path on the maze as they figure out the respective code blocks and trajectory for the robot.* The graphic image of the maze provided an opportunity for the student to use his perceptual and motor senses to map the path the robot would travel onto her body thus embodying the path.

Overall, this vignette provides evidence that learners used language representations, symbols, and embodiment to help them add and iteratively express CT practices mediated by the maze, which afforded a visual guide for students to count squares, convert distances into time, and subsequently decompose, identify recurring patterns, and use loops to program.

Discussion on Graphic Representations

The monorail path used in this activity provided a form of graphical visualization that helped learners visualize and abstract ideas such as calculating distances as a function

of time. In this example, learners successfully navigated the challenge of understanding the relationship between speed and time by establishing a direct correlation between distance and time. The grid squares on the path served as a visual aid for distance representation, allowing learners to abstractly equate each square to a consistent distance and consequently time, effectively leveraging the invariant property of speed.

Graphical representations designed into the maze helped learners to decompose the task by making stops, identifying patterns on the path, and designing loops on their algorithm. I argue that the use of graphical representations can be highly beneficial in CT environments. As shown in the example, graphic representations can support young learners to understand complex concepts through abstractions that can lead them to understand relationships between variables, such as the covariant relationship between speed and time. This vignette provided an opportunity to understand how seamless relationships between concepts and sometimes disciplines can be translated using representations, in this case across CT and mathematical concepts. The students used graphic representations, in association with language, symbolic, and embodied representations to engage in CT components of decomposition, pattern recognition, abstraction, and algorithm design.

Language Representations

Description of Language Representations

In this work, I define language representations as the use of utterances or writing to convey, discuss, and reason about CT concepts. Language representations I

use for this study include verbal explanations, instructions, descriptions, or discussions related to the activities. For this work, I argue that language representations can support understanding, communication, and collaboration among learners (Alibali, Bassok, Solomon, Syc, & Goldin-Meadow, 1999). The students and teacher I observed in this study, engaged with language representations by listening, speaking, reading, or writing about computational tasks and concepts. They used spoken language to verbalize their thoughts, explain reasoning, ask questions, or discuss ideas with others (National Research Council, 2010). Jeannette Wing's definition of CT supports the idea that to think computationally involves the capacity to express ideas in computable form. This involves constructing representations using highly constrained computer languages, which, unlike natural languages, offer limited perspectives, abstractions, and articulation capabilities.

There is evidence in the literature that novice computer users typically begin incrementally refining imprecise mental representations expressed in natural language. When they verbalize this discourse, they support tangible semiotic transformations that lead them to gradually build precise expressions that can evolve into computable code fragments, blending with natural signs. This process supports the development of novices' semiotic understanding, enabling them to compose larger, meaningful, and executable computer programs (De Souza et al., 2011). For this study, I use verbal expressions the learners engage with to understand the representational relation between language representations and computational tasks:

Supporting understanding: Language representations help learners construct and refine their understanding of the activities and CT concepts by providing a medium for expressing, describing, and reasoning and connecting the real-world framing of the problem to abstract ideas.

Facilitating communication and collaboration: Language representations can enable learners to communicate their ideas, questions, and solutions to others, fostering collaboration and promoting the exchange of diverse perspectives and strategies.

Enhancing metacognition: Supporting learners' semiotic transformation and facilitating the creation of more complex and abstract machine executable programs more constrained from natural language representations.

Grounding understanding of CT: In this study, the teacher engages learners using language to explain the sequence of actions, that their algorithm employs to guide the robot's movements. By engaging with language representations (spoken), students practice articulating their thought processes and reasoning, which helps them decompose and refine their understanding of CT tasks and enhance how they reason about and with CT. Additionally, discussing their algorithms with peers can help them promote collaboration, expose them to alternative strategies, and support the development of problem-solving skills. Below is a vignette script I selected to foreground language representations.

Vignette Foregrounding Language Representations

The vignette below follows the teacher as she introduces the lesson to the students. This is a particularly short vignette, and I chose this vignette to demonstrate how the teacher uses explanations and prompting to frame the activity for the students to work on. The activity is framed around the idea that many parks are big enough that it would require a lot of walking to get from one attraction to the other. One way to move visitors through a park efficiently is by using a monorail. Based on how the teacher framed the activity, the learners were encouraged to use different blocks the teacher described to program stops for the monorail. Some of the focal group students engaged in the activity by writing down the steps of the program in their notebooks, discussing ideas with their peers, and then building the algorithms (Figure 35).

Vignette Script for Language Representations

//Researcher observation: The teacher starts by framing the question, discussing the activity and describing blocks needed to program the robot to navigate the monorail.

Teacher: Okay we're on lesson 8, the monorail stops. Remember all of our Sphero activities have to do with an amusement park, what might be a monorail at an amusement park? Has a rail in it? what do you think?

// Researcher observation: Teacher calls a student to answer

Student M: I think they use a monitor to control the parks

Teacher: oooo, so I like that you are thinking of a monitor because you are looking at monorails and looking at parts of the words; anybody else?

Student T: I see rail

Teacher: You see rail so what does rail mean usually?

Student T: a roller coaster's rail?

Teacher: Okay, so if you've been on a roller coaster in an amusement park you'd see it has a rail that the cart follows right? (stretching her arms in front and mimicking the movement of a train Figure 36).



Figure 36: The teacher stretches her arms to mimic the movements of a train.

Hmmm, One last prediction.

// Researcher observation: Student spoke inaudibly.

Teacher: Here is a monorail!!! (Clicking on the screen to reveal the image of a monorail), (Figure 37).

Right there!! Do amusement parks have monorails?

Class: Yes



Figure 37: The teacher shows the image of a monorail to students.

Teacher: And what's the purpose of the monorail? What do they have monorails for?

Student A: Roller coaster?

Teacher: That kind of not look like a roller coaster, or if it is, its a very boring roller coaster (laughs) what do you think of that prediction, what do you think K?

Student 4: I think its so people don't have to walk a lot

Teacher: its to transport people right? Because have you been to some amusement parks? Its so big it takes a really long time to walk from one end to the other, well a monorail can get them from one place to the other place really…

Class: Fast

Teacher: Really fast, really quickly, so, many parks are big enough that it would require a lot of walking to get from one attraction to the other. One way to move visitors through a park efficiently is a monorail. Mono meaning one, rail meaning one rail.

//Picking up one of the grid maps she had designed the teacher then continues; "your mission is to figure out a code we can use in our park. Here's a map of our monorail path,I actually did a little bit different one and its yellow (on the screen) so you cannot see the path". Holding up the map of the monorail path she asks the class "can you see the path? that we want a monorail to travel?

Class: yes

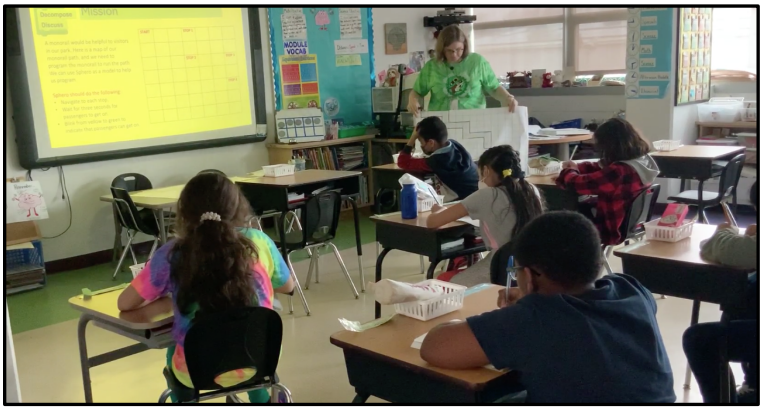


Figure 38: The teacher holds up the map of the monorail and projects the image of the map on the board.

Teacher: we're gonna need to program the monorail to run this path and Sphero can help us test it out. Sphero should do the following: Navigate to each stop, that means. What do you think navigate to each stop means? What do you think? Calls out student F.

Student F: Inaudible

Teacher: So its gonna move, its gonna move from one stop to the next., so its gonna navigate to each of the 6 stops. What will it do when it gets to each stop? It will wait for 3 seconds for passengers to get on. Its gonna sit and wait for passengers to get on. It will blink from yellow to green, to indicate that passengers can get on. So lets write those down inside of our notebooks so we know what we need to program Sphero to do. Navigate to each stop, it needs to go, travel to each stop along the path, wait for three seconds and blink from yellow to green to let the passengers know they can get on.

Students look at the map on the screen and write down the algorithmic steps into their notebooks (Figure 39a).

Figure 38. Teacher Shows the map of the monorail path to students while they write down algorithmic steps to complete the path into their notebooks.

Teacher: Now we know what we need Sphero to do, what blocks will we need? We need blocks to navigate, we need blocks to wait, we need blocks to blink right? Okay, so what blocks do you think we should use? words used to program Sphero. So now we know what we need Sphero to do, what blocks will we need? We need blocks to navigate, we need blocks to wait we need blocks to blink right? So, what blocks do you think we should use? So we might need to do some measurements to figure out what?

What M? (prompting a student)

Student M: To see the length and width.

Teacher: The length and width of what? How do you know the length though? Calls out Student A

Student A: The whole system

Teacher: The whole thing?

Student A: Yes

Teacher: Maybe; how about you and your group try to figure out what is the length that you are gonna need to know so we need to enter something right her but what block would we put that measurement in? You think which block we'd put that measurement in? All right that's okay. (Calls out student N)

Student N: Distance?

Teacher: So somehow, we need distance right? We'd talk about distance in a second. What other things do we need Sphero to do?

Student K: To turn from yellow to green.

Student A: To stop and wait for 3 seconds.

Teacher: Okay, is that the only blocks we need? To roll stop, blink?

Student: The Speed?

Teacher: The speed: So that is what we put in to tell us the distance right? What else do we need Sphero to do to complete this path?

Student: The heading

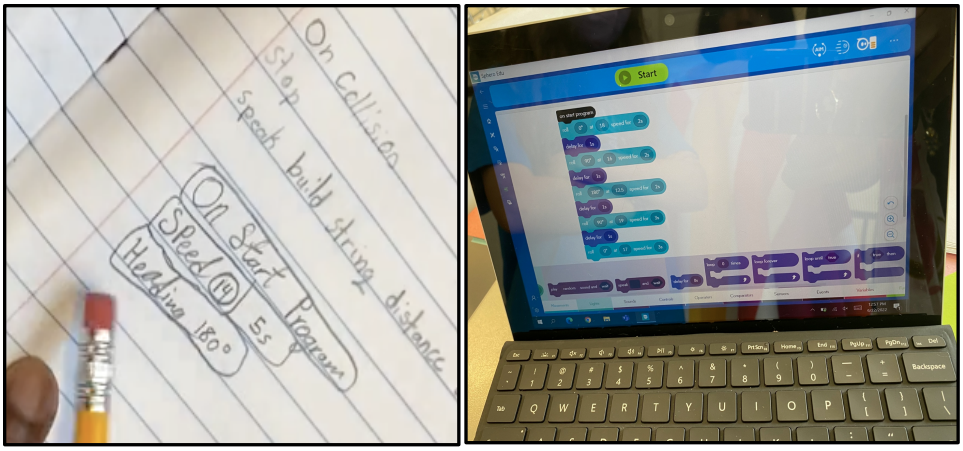
Teacher: What's the heading?

Student: The turn

Teacher: We need Sphero to turn right? So, we're gonna need to adjust that heading and that is for you and your teams to figure that out. We need the roll block. That is the very first one we used, and we hand heading zero the whole time but what's the other thing we changed in the roll block? The speed and the…? Because the roll block will have to go a certain distance and distance needs speed and what? I remember when you are in the roll block, you are gonna adjust. Today we've changed the heading to make a turn, but just to make Sphero roller need what? If we just put the Speed there, it won't go anywhere. What else? Speed and what?

Student A. Time.

Teacher: Speed and Time



(a) (b)

Figure 39: (a) Students translate language representations through writing to communicate the movements of the robot on the monorail path. (b) Language representations supported learners to the use of block programming language to write instructions for the robot to follow.

Vignette Analysis for Language Representations

In this vignette, the teacher used language representation to discuss the activity in ways that prompted the learners to think about the task. For example, framing this activity around lived experiences provided pathways for the teachers to engage learners to think about the implications of the processes and how different parts of systems function. The teacher begins by saying; *"remember all of our Sphero activities have to do with an amusement park, what might be a monorail at an amusement park? Has a rail in it? what do you think? Student M responds "I think they use a monitor to control the parks."* The teacher responds, *"oooo, so I like that you are thinking of a monitor because you are looking at monorails, you're looking at parts of the words; anybody else?"* Student M's response supports the importance of prompting students to support their understanding of contexts. From student M's

response, he/she does not immediately connect to the context of what a monorail would mean in an amusement park.

When there is a disconnect in context, students can sometimes find it difficult to understand the problem. This is seen when student T answers saying *"I see rail"*; the teacher further prompts; *"You see rail so what does rail mean usually?... a roller coaster's rail?"* The student is making associations between the word "rail" and activities in the park that may likely use rails, indicating their connection to prior knowledge of where rails can be used in an amusement park. The teacher supports student T's thoughts saying, *"Okay, so if you've been on a roller coaster in an amusement park, you'd see it has a rail that the cart, the car follows right?"* (stretching her arms in front and mimicking the movement of a train as seen in figure 36). The teacher provides context for the activity by connecting the activity with real-world examples provided by the framing of the Sphero.Math curriculum. Through comprehensive exploration using natural language, the teacher breaks down the activity sometimes using analogies and metaphors. This breakdown of the activity expanded learners' metacognitive ability to express themselves in computable language by helping them to construct representations of real or imaginary objects using constrained languages that would later be translated into computer languages that they would enter their programs to control the robot (Figure 39).

Additionally, language representations were used as a first-step strategy to organize tasks. This was seen when the teacher introduced the programming activity by simplifying and verbalizing the steps needed to program saying; *"now we know what we need Sphero to do, what blocks will we need? We need blocks to navigate,*

we need blocks to wait, we need blocks to blink right? Okay, so what blocks do you think we should use?" The teacher further guided the students using verbal explanations to provide students with an understanding of the different blocks required to complete the task. The teacher emphasizes the definition of each block about the action the block will perform; For example, *"blocks to navigate",* meaning the roll block, *"blocks to wait"*, meaning the delay blocks, *"blocks to blink"*, meaning the event (light) blocks. Language representations in this case are used to decompose the steps into simpler understandable steps in ways that learners could easily translate abstract physical concepts into executable algorithms.

Furthermore, the teacher emphasized the importance of measurements in the task, elucidating that students must *"count the blocks"* on the monorail path to determine specific lengths to accurately program the Sphero robot. She articulated this point by stating*, "So counting the blocks is going to be the measurement we need, some measurement to figure out the length and then the distance?"* To this student, M answered *"Distance"*. In emphasizing that the students need to have the distance value established first to understand which variables, to manipulate for speed or time, the teacher uses language representations to support the students' decomposition and find patterns by counting, squares, and eventually finding variables to build algorithms.

The teacher and learners used language representations to structure the activities by breaking down the steps involved in the tasks, identifying instances with patterns, and organizing and structuring the language to use towards building the algorithm. These language representations expressed by the teacher, were used to

build scaffolds that helped students translate their understanding of how to translate between “distance” to “roll block”. For young learners, the roll block if not properly explained, may not immediately translate into the idea that speed and time can work as covariant to calculate the distance. The teacher through these language representations supported learners to construct constrained languages needed to translate information from the physical to the virtual space.

This vignette provides evidence that novice programmers can benefit from navigating between imprecise natural language representations, sometimes expressed in informal discourse to highly constrained programming language. This process can involve using natural language to describe actions and processes or creating tangible signs like gestures that can aid semiotic transformations, eventually leading to the composition of formal and precise expressions for building computable code. This vignette illustrates how language representations can translate into CT practice through decomposition, pattern recognition and eventually building algorithms from natural language that is gradually constrained into code.

Discussion

The teacher in this vignette uses language representations to build context and help students break down and interpret abstract concepts of obtaining distance through the covariation of speed and time. This way of using language to support learners' interaction with ideas in computable formats was achieved using natural language constructs to explain the constructs of computer languages. Moreover, language representations in this activity helped provide pathways for the teacher to

contextualize, sequentially organize and verbally communicate the steps students needed to build an algorithm to control the robot. This vignette highlights the significance of leveraging language representation as a means for explicit instruction in ways that can foster active teacher/learner engagement that can support students' CT practice understanding. In teaching CT activities to young learners, there is room for investigations and explanations to check where learner's familiarity or understanding of the activity may be. Through this activity, I was able to understand that one advantage of teaching CT to young learners in curricular contexts is the opportunity it provides to use language representations and explanations to support learners in understanding concepts in ways that can support their transition from natural language to constructing more constrained computer readable programming languages.

Symbolic Representations

Description of Symbolic Representations

Symbolic representation refers to the use of symbols or signs to represent concepts, objects, or relationships in a structured and meaningful way (Wearne & Hiebert, 1988, p. 372). These symbols can be letters, numbers, mathematical operators, or any other characters that have a specific meaning within a particular context. Symbolic representations are commonly used in mathematics, science, and programming languages, where precise and unambiguous communication is essential. In mathematics, for example, symbols like +, -, x, and / are used to represent basic arithmetic operations. Similarly, in programming languages, symbols are used to

represent data types: (integer: int, string: (str), operators: (addition+; subtraction -; multiplication *), functions: (print: print(), Input: input() (Moore et al., 2020).

In certain contexts, programming languages may incorporate a fusion of language and symbolic representations (Bau et al., 2017). For instance, block languages seamlessly combine both linguistic and symbolic representations (Lin & Weintrop, 2021; Maloney et al., 2010). The block programming language used in the Sphero.edu app interface is designed to provide visual cues to users on aspects such as the block's shape, color, and use of natural language inscribed on the blocks to communicate their function and break down vocabulary in ways that can directly constrain the language for direct manipulation into the code.

Unlike text-based languages, block programming languages are intricately connected to their programming environments. Almost all block environments incorporate specific interface conventions to address essential usability issues. Block-based programming environments can handle vocabulary by grouping blocks into categories based on functionally related palettes designed to be visualized on the screen (Bau et al., 2017). For example, blocks can support novice programmers in deciphering between expressions versus commands versus declarations by visualizing the shapes of the blocks (Lin & Weintrop, 2021; Maloney et al., 2010). In the Sphero.edu programming environment, commands are designed to snap together like puzzle pieces with nubs and notches that snap together whereas expressions take on sleek forms that seamlessly slot into matching openings. All these attributes emphasize the symbolic nature of block programming languages. However, I describe how the design of the blocks and the programming environment reduces the cognitive

load for young inexperienced programmers as described in Chapter Five. In CT programming environments, symbolic representations can support learners navigate tasks in the following ways:

Abstraction: Symbolic representations can allow learners to abstract complex concepts and ideas into simplified symbols or notations, making it easier to understand and manipulate information.

Problem Decomposition: Symbols can help learners break down complex problems into smaller, manageable parts. This facilitates the application of CT principles such as decomposition allowing learners to focus on individual components.

Pattern Recognition: Symbolic representations can sometimes help highlight patterns and relationships within data or algorithms. This can support learners in recognizing recurring structures and understanding the logic behind them.

Algorithmic Thinking: Symbols can be used to represent algorithmic steps, supporting learners in developing sequential and logical steps to build functioning algorithms or approaches to problem-solving.

Communication: Symbolic representations can provide a standardized and concise way for learners to communicate ideas within a community of practice. For example, in mathematics standard symbols are used to communicate ideas and notions. This can be crucial in collaborative learning environments when sharing ideas or solutions with others.

Visualizing Data: Symbols can help learners visualize and interpret information. For example, in block programming environments, blocks as symbolic representations can provide visual cues to users on aspects such as the block's shape, color, and use of natural language inscribed on the blocks to communicate their function.

Translating Between Spaces: Learners can use symbolic representations to bridge the gap between virtual and physical spaces, translating concepts from one context to another. For example, abstract concepts from the virtual world can be translated into tangible elements in the physical world using symbolic representations. Such symbols may sometimes bear resemblance to real objects they symbolize or could be simplified or stylized versions of objects they represent, making them easily recognizable and understandable. This use of symbolic representations is mostly seen on block programming platforms that use symbols and sprites to represent real objects in ways to support young learners in easily recognizing and associating them with real objects.

Vignette Script for Symbolic Representations

Student 1. Yeah

Researcher: Each function is represented by a hashtag.



(a) (b)

Figure 40: (a) Five full function codes to represent each of the five activities in the amusement park. (b) Five function codes represent five activities for the amusement park.

Researcher: so you have like like, one two three four five functions?

Student 1: yes

Student 1: The first hashtag is here; (pointing to a hashtag representation on their design) and the blue light is facing the red dot

Researcher: so the dot is "aim" (Red dot on the activity board) (figure 41)

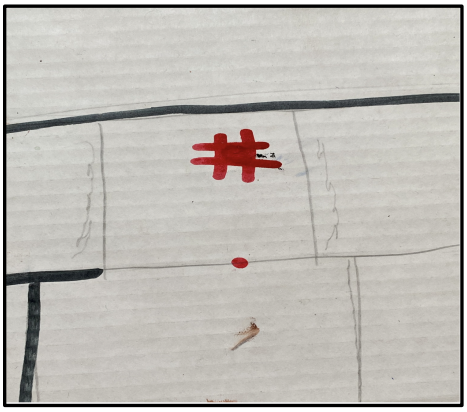


Figure 41: Dot and Hashtag symbols were designed for the activity.

Student 1: Yes

Researcher: let me see the code, where is the main code, the big mother code? So what does this one do? (Referring to the code on the ipad the students designed.

Student 1: That is the maze code, it deals with the maze, so when it starts here. (pointing to parts of the activity on the mat), It goes, here, here here (pointing to a shaded area of the activity

Researcher: and then, the next one?

Student 1: This next one is the water park, the water park is supposed to be facing this way, (moving over and pointing to the opposite side of the activity-colored blue to represent water) and then it just makes a sound because we don't know what to do for that. me: so why do you use a function for that?

Student 1: So we don't, so we separate things

Student 2: So, the idea is that you can choose which one they want to play

Student 1: So, they are not confused like, so I can put this sound like the restaurant or something, it just makes it easier.

Research: and then the next one? The next function?

Student 1: The next function deals with the roller coaster, it just goes ruueee

Researcher: Okay.

Student 1: This next one is the restaurant, all it says is yum yum yum, and then this last one is the one that goes.

Student 2: To the beach

Student 1: From here, (moving his fingers on the activity)...hmmm hmmm hmmm, splash splash splash

Researcher: Okay, so now tell me, does this mean we can put this function on any place and its gonna do something?

student 1: No, because that's why there's the instructions (picking up a piece of note paper on which they had written out color-coded instructions) and the hashtag marking system as you.

can see the hashtags are color-coded which you can see by the colors here, (showing the blue, brown and orange color code system on the piece of paper) and then reading the color code manual; brown (water park).

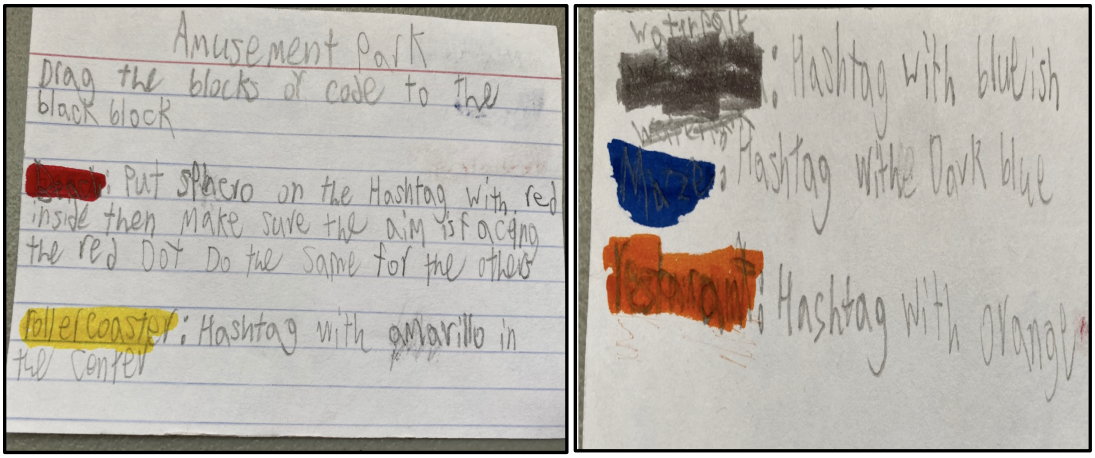


Figure 42: Color-coded instruction manuals designed by students to help visitors easily navigate the amusement park.

Researcher: How did you come up with the idea of having colors?

Student 1: So, people wouldn't get confused (student two breaks in saying "yeah")

Student 3: I think they'll also use their imagination and it also be fun to add some colors because it could also be errmmm representative of spots and y'ahh

Researcher: so why did you decide to use hashtags? how did hashtags come about?

Student 1: I just put them there to be easy so people don't spend too much time with how am I gonna make a way? so it just, its ....

Researcher: Okay, nice, let's run this

Student 1: Now let's drag the beach (pulling the functions icons on their iPad) the most command.

Researcher: the most complicated command?

Student 1: yes

Researcher: complicated in what sense?

Student 1: As in… then runs the command and a swooshing beach sound was heard from the iPad, including a roll unto the part of the activity designed to represent a beach the student then selected the next function, placed it on the dot, and the hashtag and then ran the code.

Student 1: There, it now works (Sphero rolls from the hashtag right onto the beach) he then continues saying "and then we were trying to make it go back here, roller coaster, water park comes to the maze lets turn back and then go to the, as all in one park code, but then it won't be very interactive.

Student 3: So now we're going to go to the restaurant; she selects the function restaurant on the screen Student 2 places the Sphero robot on the hashtag representing the restaurant following their color code scheme and then says (inaudible)



Figure 43: Five hashtags and dots represent the five activities designed for the amusement park. Hashtags and dots were used as symbolic representations for each function and the robot's taillight.

Student 1: aim it (while student three aims the Sphero using the virtual aim)

Student 1 moves to the activity board and gives directives as to the positioning of the aim taillight saying " the other way, no, there" making sure the taillight is aligned with the red dot the had made on the activity to represent the aim, or direction for heading the Sphero will move. They in these instances were making virtual representations both from the application icons and the robot into the physical space.

Student 3 then shows two other functions that do not require any aiming saying "these two do not require any aiming because it is just the sound" meaning the Sphero won't make any displacements hence no need for the headings.

Researcher: So, they don't require any aiming because there is just the sound? so why did'nt you make a move from the beach to the restaurant?

Student 1: Because that will be too hard (indicating the challenge of finding a coding pattern

Researcher: Because it has to go in a circle?

Student 3: everytime it would like bounce on the wall

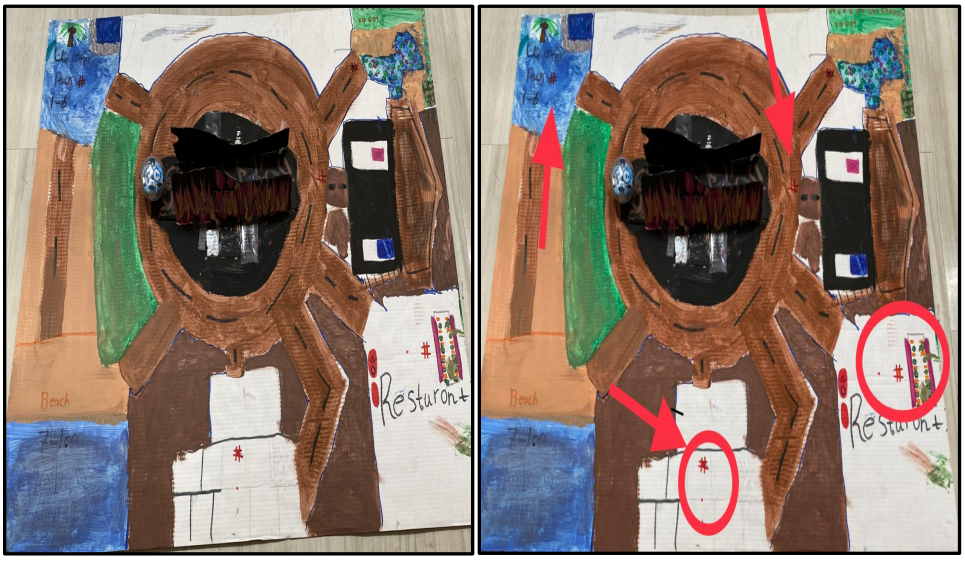
Student 1: yeah

Researcher: I thought you said you knew how to get Sphero make a circle? So you would have made like really sharp turns?

Researcher observation: The maze, the roller coaster had dots associated with the hastags. The beach and the restaurant did not have any red dots associated with the hastag because Sphero only made sounds yum yum yum representing eating and swooshing beach sounds.

Explanation of Vignette

In this vignette, students used symbolic representations to help them organize and build programs to control the robot navigate amusement park activities they designed for a final class activity. They used symbols such as dots hashtags and functions, as seen in Figures 44 and 40 to design a series of five activities divided into five functions which they program to each represent a specific activity.



(a) (b)

Figure 44: (a) Amusement Park activity is modeled to represent amusement activities for visitors who visit an amusement park. (b) Hashtags, red dots, and color codes are embedded into the activity map to help guide visitors to navigate the park with ease.

I present an analysis of how students used symbolic representations and other forms of representations to navigate the activity they designed. First, the students started by breaking down the activities into parts using symbolic representations. As seen in Figures 40 and 43, the students use decomposition to breakdown the activity. They break down the activity using symbolic representations such that each park activity is represented both in the physical space by a hashtag (Figure 43 and 44) and in the virtual space by a function (Figure 40). The researcher asked the students the following questions after looking at their design; *"so you have like, one two three four five functions… each function is represented by a hashtag?"* The student answered in the affirmative and further went ahead to explain the role of the hashtags in their design. Student 1 explained; *"The first hashtag is here (Figure 41); (pointing to a hashtag representation on their design) and the blue light is facing the red dot"(Figure 45).* The researcher further prompted the student to understand the meaning of the dots on the design by asking; *"so the dot is "aim"* and the student responded *"Yes"*. The students created and designed their own symbolic representations of the "aim" light into the activity.

The students are creatively taking representations from the Sphero robot, that is, the blue taillight "aim" used to indicate the heading or direction of the robot, usually called "aiming" from the robot to the activity they are designing (figure 45). The dot represented a symbolic representation that attended to the robot. This symbolic representation helped the students visualize the robot's orientation and direction the robot will roll.

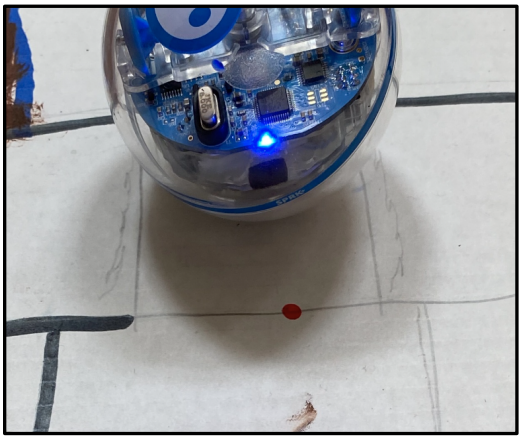


Figure 45: The students use a blue taillight from the Sphero robot to represent the robot's heading or direction, which they call "aim."

As seen in this vignette, students use hashtags to represent functions on physical activity (figure 44b). The students translate representations from the virtual space (functions) into tangible visual elements in the physical space (hashtags). This kind of symbolic representation translated from virtual 2D to 3D physical spaces can be argued to be enacted as a form of abstraction to break down the complexity within the activity. Learners used CT components of decomposition and abstraction to consequently make the activity easier and less complex to navigate. Student 1 further explained how the activity works and how they have programmed the hashtags using functions:*" That is the maze code, it deals with the maze, so when it starts here…"* and further, pointing to parts of the activity on the mat, he said, *"it goes, here, here here"* (pointing to a shaded area of the activity). These pointing gestures are used to direct attention to the details of the path the robot will follow saying "here, here. Here.." recognizing there's a pattern in the path.

Furthermore, the researcher asked the student to explain some of the patterns he was presenting, *"and then, the next one?"* Student 1 explained; "*this next one is the water park, the water park is supposed to be facing this way, (moving over and pointing to the opposite side of the activity-colored blue to represent water) and then it just makes a sound because we don't know what to do for that"*. The researcher further prompted, *"so why do you use a function for that? "*Student 1 answered; *"So we don't, so we separate things"* Student 2 further reinforces the idea by saying; *"so the idea is that you can choose which one they want to play"*, student 1 adds; *"So they are not confused like, so I can put this sound like the restaurant or something, it just makes it easier"*. The student's language choice, expressing the idea of *"separate things", "choose which one they want to play, "makes it easier,"* implies a deliberate design decision to decompose tasks within the activity. This provides evidence that the students are consciously decomposing the activity by using functions for each component of the tasks. This approach enables park visitors to select and interact with specific attractions of their choice. This reflects their ability to break down abstractions in a CT environment using symbols.

This vignette shows that learners relied heavily on symbols both in the physical and virtual spaces. Given the prevalent use of symbols in this vignette, I provide further evidence of how symbolic representations supported CT practice in the next section.

Symbolic Representations and Computational Thinking

This vignette provides evidence that there was a natural alignment between the use of symbolic representations and CT practices leading to a possible hypothesis that, creating, and using symbolic representations could constitute a form of CT practice. Below I provide further evidence of the fusion between symbolic representations and CT practice.

Decomposition: By dividing the overall water park experience into separate, smaller components (activities or attractions) represented by hashtags, functions, and dots, the students are breaking down a complex problem into manageable parts, using symbolic representations to navigate the physical and virtual spaces. This is evidenced when the researcher asked the question; *"so why do you use a function for that?".* Student 1 answered; *"so we separate things".* This choice of decomposing the task into parts enabled the students to focus on designing and programming individual attractions, making the task more accessible and understandable for the students and providing a simplified user experience for the users of the park.

Abstraction: The use of functions to represent each activity is an example of abstraction. As mentioned above functions in this study are classified as symbolic representations. This is evidenced when the researcher asked the question; *"Each function is represented by a hashtag; so, you have like, one two three four five functions"?* And the student replied *"yes"*. The student further provides more evidence to this when the researcher asked*; "so why did you decide to use hashtags? How did hashtags come about?"* Student 1 replied; "*I just put them there to be easy,*

so people don't spend too much time with how am I gonna make a way?" Here, symbolic representations are supporting learners to create abstractions to navigate the task. The student's choice of using abstractions through symbolic representations, where specific details of the code are hidden, and only the necessary information is exposed, suggests that the students have gained experience and are expressing mature symbol user abilities.

This level of abstracting using symbols shown by the students portrays a certain level of proficiency in programming that allowed the students to decompose the tasks and focus on a high-level design process of the water park experience without getting overwhelmed by the intricacies of each attraction's implementation. There is evidence in this vignette that proficient symbol users exhibit comparable fluency in using symbols as they do in navigating programming syntax.

*Modularity: T*he students' choice of using functions for each activity demonstrates their understanding of modularity, a key concept in CT. Modularity in this case refers to the idea that the students are designing the system as a collection of independent, decomposable components. This is seen during their execution of the code as student 1 says: *"There, it now works* (Sphero rolls from the hashtag right onto the beach) he continues saying *"And then we were trying to make it go back here, roller coaster, water park comes to the maze let's turn back and then go to the, as all in one park code, but then it won't be very interactive.* Student 3 reinforces this evidence when she says, *" so now we're going to go to the restaurant";* she selects the function restaurant on the screen. By designing each attraction as a separate module (function),

the students were able to easily modify, add, or remove attractions without affecting the rest of the system, a key concept in the engineering design process.

Proficiency in the use of Symbolic Representations and CT can Lead to Creativity

The students' fluency in symbolic representations enabled them to surpass mere competence in engaging in components of CT practices. I observed that students were able to extend their abilities to encompass user perspectives by designing personalized and interactive experiences. The students added creativity to their work by creating color-coded instruction manuals for visitors to the amusement park (figure 42). Student 3 *thinks they'll also use their imagination, and it will also be fun to add some colors because it could also be erm representative of spots and y'ahh".* She further explains the role of the instruction manual saying; *"No, because, that's why there are the instructions* (picking up a piece of note paper on which they had written out color-coded instructions) *"and the hashtag marking system as you can see the hashtags are color coded which you can see by the colors here",* (showing the blue, brown and orange color code system on the piece of paper) *and then reading the color code system; brown water park.* This is evidence of how the students were able to explore the potential of symbolic representations to connect to creativity beyond just the given tasks. Students designed symbolic representations for the virtual and physical environments and the users by providing user experiences designed for users.

Through this vignette, I have been able to demonstrate evidence that proficient symbols users like the students in this focal group are also as proficient in designing

and interacting fluently with the programming environment and they can extend their interactions beyond CT practice to creativity and practical user-centered design experiences.

Discussion

To conclude, the students demonstrated through this concluding activity that symbolic representations offer avenues for practicing CT skills, including decomposition, modularity, and abstraction. The students' amusement park activity effectively supported the students to express their CT understanding using functions, and color-coded hashtags to create engaging and interactive experiences for park visitors. The complexity and interactivity evident in the designs created by students in this concluding activity highlight the effectiveness of the Sphero.Math curriculum. It successfully facilitated the use of symbols that translated to the acquisition of CT and programming skills.

Students created and used conventional and novel forms of symbolic representations, and assigned meanings to them within the contexts for which they were used. For example, students used hashtags to represent functions and each function represented an activity in the amusement park. Hashtags are typically used on social media platforms to categorize content and make them discoverable by a wider audience. In this context, the learners assigned a novel symbolic meaning to hashtags, that of representing amusement park activities. Likewise, students used dots to represent the taillight (aim) on the robot. Conventionally, in writing, dots are often used as punctuation marks, such as periods (.) to end sentences, ellipses (...) to

indicate a pause or omission, and decimal points in numerical notation. In this case, the learners assigned a new meaning for dots, to represent the taillight (aim) of the Sphero robot.

This is evidence that representations can support learners within a shared community of practice to develop language around tasks that can be conventional or cultural and only shared by a group of users within a specific task (community). Learners navigated between using conventional and shared symbols as they used conventional symbols to color code areas on the activity map, which they used to design a manual for human users. In most robotic learning environments, information is usually translated from the physical to the virtual space through visualizations; that means collecting data from the physical environment and translating them into formats that the computer can understand. The students in this activity reversed the information process providing representations from the virtual space onto the physical space by translating machine-readable information into human-readable formats.

Embodied Representations

Description of Embodied Representation

Attending to gestures in multimodal environments can be particularly productive. In mathematics, gestures as representations can take on a wide variety of forms, and navigating those forms of gestures becomes a crucial component of the learning environment (Hostetter & Alibali, 2010; Nathan et al., 2017). This work has led me to consider the potential of bringing an embodied perspective to research on the interactions that occur in technology-enhanced multimodal learning environments

(Kirsh, 2013; White & Pea, 2011). In CT environments, embodied representations can often be afforded due to the nature of the activities that involve hands-on activities, such as manipulating robots, constructing physical models, or interacting with tangible interfaces. It has been argued in the literature that embodied representations can be effective learning tools in CT environments, as they can support the development of problem-solving skills, foster a deeper understanding of concepts, facilitate communication amongst peers, and transfer knowledge to real-world contexts (Fofang et al., 2021; Shokeen et al., 2020). Young learners tend to navigate embodied representations by actively engaging with the physical environment (objects and tools) using their bodily experiences to think through and make meaning of abstract concepts. In this section, I use the following aspects to explore the relationship between embodied representations and learning:

Supporting cognitive development: By connecting abstract ideas to bodily experiences, learners can intuitively engage parts of their bodies with activities in ways that can facilitate abstraction and problem-solving. For instance, when students physically manipulate objects or engage in hands-on experiments it encourages them to explore and connect theoretical knowledge to practical, real-world applications in ways that can strengthen their cognitive understanding of concepts.

Enhancing motivation and engagement: Embodied representations may increase learner's motivation and engagement by providing learners with agency and support to interact with the complexities of the learning environment (Fofang et al., 2021).

Engaging with embodied representations has the potential to make learning more meaningful, which can lead to better retention of ideas and concepts.

Communicating and connecting learning experiences with lived experiences: Embodied representations can support learners in communicating amongst themselves and connecting their ideas to real-world contexts by connecting abstract concepts to their lived experiences (Shokeen et al., 2020). This connection can help learners recognize the relevance and applicability of concepts in their everyday lives and other domains.

Description of Activity Using Embodied Representation

The example of the activity below with students using embodied representation in a CT environment to learn is based on a task where students program a robot to navigate a triangle. In this classroom-based activity, learners are expected to program the robot to engage with the environment, adjusting the robot's movements, observing the outcomes, and refining their algorithms based on their physical experiences. By working with the robot and the physical environment, students use iterative logic to refine their thinking, decompose the tasks, and combine embodied experiences to program. Learners relied on embodied interactions to deepen their understanding of angles to enhance their programming skills. Learners' use of embodied practices proved to be a valuable form of representation to explore in CT learning environments. The activity involves modeling triangles to clean structures in an amusement park. This framing is built around the idea that triangles

can give strength and stability to structures. This activity is listed in lesson #9 on the Sphero.Math curriculum as triangles (Figure 46). This activity aligns with the common core state standards in mathematics and standards for mathematics practice as shown in Appendix A.

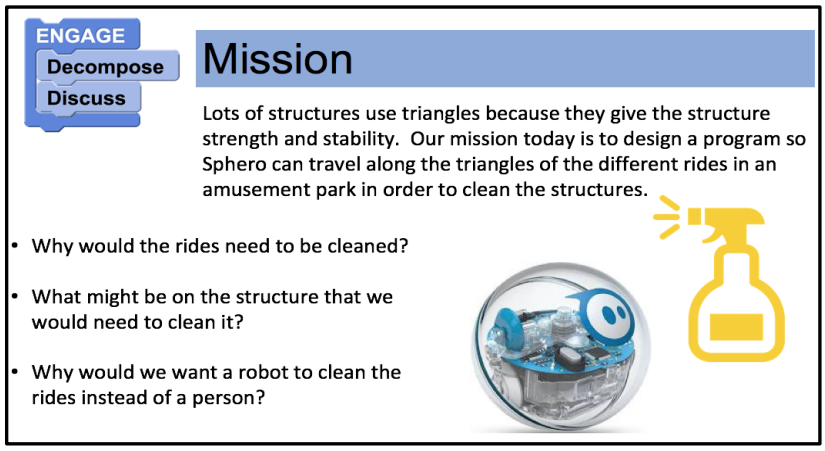
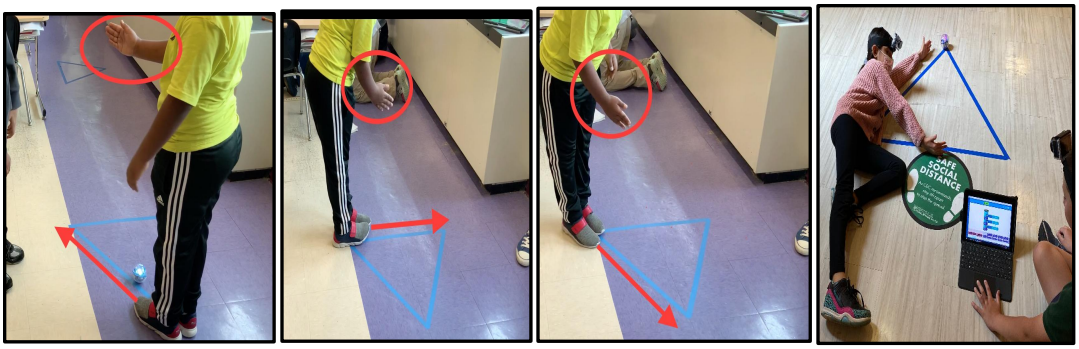


Figure 46: Sphero.Math activity model for Triangles.

For the concrete model, teachers used masking tape to design equilateral triangles on the floor (Figure 47). Students used the Sphero robots and other materials such as meter rulers, protractors, and sometimes their bodies to problem-solve the tasks.



(a) (b) (c) (c)

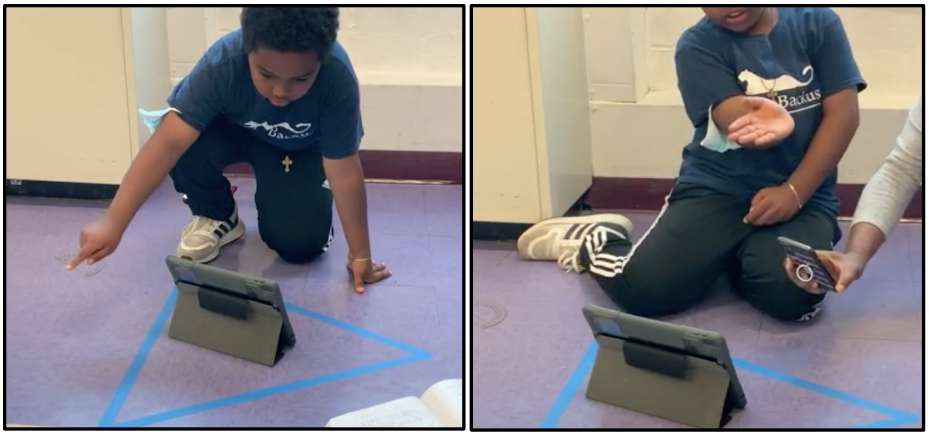
Figure 47: Students use their bodies to understand abstract concepts like angles (a, b, c, d). Student walks around the triangle with his right hand in front of him, to embody the triangle's path (a, b, c). Student embodies one side of the triangle by spreading her body to mimic an angle.

Vignette Foregrounding Embodied Representation

The vignette provided captures instances where students use their bodies and a combination of other tools in the environment to interact with concepts, and ideas, and refine their thinking about angles and directions the Sphero robot should roll on the triangle. This vignette involves a focal group made up of 3 students, two boys, and a girl, working together to program the Sphero to roll on the sides of an equilateral triangle path. In this vignette, I report observations and answers to questions the students were asked from the beginning of the activity as they worked through the challenges of programming the robot to navigate the sides of an equilateral triangle taped to the floor (Figure 47a).

Vignette Script for Embodied Representations

// Researcher observation: At the start of the activity the student starts to program the triangle and uses his finger to point the direction he wants the Sphero to move. (figure 48a) He then explains using his palm how he expects the robot to move, indicating the angle with the swerve of his hand (figure 48b).



(a) (b)

Figure 48: A student uses both his finger and hand to show the intended direction he would like to program the Sphero to move. Notice how he curves his body to indicate the nature of the turn.

//The student proceeds to use the virtual protractor and the physical protractor to understand the angles (figure 49a).



(a) (b) (c) (d)

Figure 49: Student supports his understanding using a virtual and physical protractor: Students use pointing gestures to point to the virtual protractor for precision (b, c). Student uses a physical protractor to measure the angle (c, d).

Researcher: That's what it says (referring to the program,) its gonna continue straight so that's what it is saying here so it needs to keep turning again. So, what does it turn again to do? (referring to the robot and the angle).

Student 1: 61

Researcher: Good! Did you measure that and you are sure it's 61?

Student 2: We already measured it and it all gave 61

Researcher: Because it is equilateral right?

Researcher: aim it properly so it goes on a straight line.

Student 1: So, make it this way (using his hand to indicate to student 2 the direction the aim button should align to) so it could turn that way (again using hand gestures to indicate directions for the aim)



Figure 50: Student uses gestures to communicate with his teammate the direction the robot should move.

Researcher: What you want to do most time is you want to try your code in parts

Researcher: Did you measure your distances and you are sure the speed and the time give that distance?

Student 1: yes

Student 1: then says: 9:44

Student 1: 270

Researcher: try it

// Researcher observation: 270 is the angle that would turn to complete the last side of the triangle, it's the angle to one side of the equilateral triangle, meaning by using his body, the virtual protractor and embodying both the robot and the triangle, the student was able to find the right angle using a combination of these representations.

Researcher: try that, let's see.

//Researcher observation: Student 1 then runs the program, the robot rolls one side of the triangle and then rolls off track, away from the other end that starts the other side of the triangle.

Researcher: So, did you see your sensor data says the angle is not sharp enough, so there is something you need to do.

The students get really confused at this point.

Researcher: So there is something you need to pay attention to, use your...use your..wait, now when you roll straight, you delay and then you have to turn, right? So if you roll straight, you turn 60, 61, ...try that and see what it does

Researcher: aim properly so we can be more precise so what will you turn again now to finish this one?

Student 1: 90?

//Researcher observation: And then one more time he steps on the triangle taped to the floor, moves his body, in the directions of the sides of the triangle with his right hand in front of him, holding the iPad with his left hand, looking into the virtual protractor on the iPad, and changing the directions on the virtual protractor as he walks on the tape representing the triangle (figure 51b).

The students run the code, and the robots make one side of the triangle and then roll away from the other tape to the opposite side of the triangle. The students are frustrated, stock and don't know what to do

Researcher: so there is something you need to fix, use your..., now if you roll straight,

//Researcher observation: When the students changed the start position on the triangle taped to the floor, they succeed in navigating two sides of the triangle, and were unable to make the Sphero navigate the third side. Faced with this challenge, student 1 starts moving on the triangle path, looking at the virtual protractor and moving the arrows of the virtual protractor to find the direction. He uses his body as an orientation to directionally find how the Sphero will move (figure 51). At this time, he's not making any kind of mathematical knowledge to find the angles. His body and the virtual protractor become the affordable tools he can rely on to map the activity on the ground, onto his body, and the virtual protractor. He regularly stops to measure the angles with a physical protractor.

Researcher's observation: This still did not work still. The Sphero kept going either straight forward or not making the angle required to make an equilateral angled triangle.



(a) (b) (c)

Figure 51: Student maps the triangle first using his right hand (a) and then starts to walk around the triangle while looking at the virtual protractor.

// At this point he's embodying both the robot and the activity and trying to find the right angle using a combination of his body, and the virtual protractor.

Student 1: then says: 9:44

Student 1: 270

Researcher: try it

//270 is definitely the angle that would turn to complete the last side of the triangle, its the angle to one side of the equilateral triangle, meaning by using his body, the virtual protractor and embodying both the robot and the triangle, the student was able to find the right angle using a combination of these representations.

Researcher: try that let's see.

//Researcher observation: The robot rolls two sides of the triangle and then rolls to the opposite side of the third side.

Researcher: see, what's the opposite of 270 now?

Researcher: check your heading, aim it properly. Did you measure the distances to make sure the speed and time correspond?

Student 1: yes



Figure 52: The student uses her hand and finger to gesture and embody both the robot and the angle.

In the vignette presented above, students worked on a CT robotics task that involved programming a robot to navigate an equilateral triangle. This vignette involved much more observations and a few questions from the researcher than conversations between team members. Data was collected by note-taking and observations so the analysis will reflect that format of data collection. In the following section, I explore different embodied representational practices that the students engaged with from the selected vignette as follows:

Finger pointing: The student uses his finger to point the direction he wants the Sphero to move *"Student starts to program the triangle, uses his finger to point the direction he wants the Sphero to move,"*(figure 48) providing an embodied representation, focusing the attention of his peers to his intended action. The student's use of pointing gestures with his finger to indicate the direction he wants the Sphero to move represents an embodied form of communication both with his peers and expressing his cognitive understanding of the task. Other than using pointing gestures to indicate directions for the robot, the students also used pointing gestures to focus

on representations and symbols like the aim light (figure 50). In this context, the student's pointing gestures served several purposes:

Supporting cognitive development: By using pointing gestures, the student externalized his thinking, making his thought process visible to other team members and possibly making it easier for him to articulate his intentions.

Communicating and focusing attention: The pointing gesture helped direct the attention of both the student and team members to the specific direction he envisioned for the robot to roll towards, facilitating shared understanding and communication. The position of the aim light is controlled from the iPad, so the hand gestures serve as a form of communication between team members for moving the "aim" taillight to a particular position.

Hand gestures: The student motioned his expectations for the robot's movement using his palm, swerving it to indicate the angle of turns *"The student started to program the triangle, and used his finger to point the direction he wants the Sphero to move. He then motioned using his palm how he expects the robot to move, indicating the angle with the swerve of his hand."*

The student's use of hand gestures, such as swerving the palm to indicate the angle of turns, is an example of an embodied representation of geometric angles. Such embodied representations found in this vignette, refer to how the student is using his body to convey, understand, or express abstract concepts (angles), or processes (positioning the "aim" taillight). This vignette supports the idea that when young

learners use robots to program in math and CT environments, they can ground abstract concepts such as angles through embodied practices.

Body orientation: Student 1 used his body to mimic the robot's movement with the help of a virtual protractor on the iPad, providing an embodied representation of the robot's path; *"student 1 starts moving on the triangle path, looking at the virtual protractor and moving the arrows of the virtual protractor to find the direction. He uses his body as an orientation to directionally find how the Sphero will move"*. The student's use of body orientation to mimic the robot's movement with the help of a virtual protractor on the iPad is another example of an embodied representation in this vignette.

By physically aligning his body with the robot's intended trajectory, the student is connecting his own spatial awareness and kinesthetic sense with that of the robot's programmed path. When the student is walking on the triangle path, he is not expressing any kind of mathematical knowledge to find the angles. His body and the virtual protractor are the affordable tools he relies on to map the directions and angles of the triangle, onto his body, and then uses the virtual protractor to find directions. The learner uses his body as an abstraction tool to understand and connect the directions of the angles to the reading of the virtual protractor.

This behavior becomes apparent as the student continuously modifies the angle values entered into the program, consistently adjusting them according to his body orientation in alignment with the directions depicted on the physical protractor. *"The student regularly stops to measure the angles with a physical protractor, probably*

with the hope that the protractor could provide him with some concrete number to enter into the code block for direction/angle". The student is likely relying on his prior knowledge and understanding of angles in a triangle. The learner tries to make sense of the difference between the value of the angles provided by the physical protractor to the value of the angles shown on the virtual protractor, by abstracting with his body. I analyze how the learners used their body orientation to ground their understanding of angles in the following ways:

Intuitive cognitive support: The students were trying to get the robot to navigate the equilateral triangle, by rolling 60 degrees for each side. They erroneously used the physical protractors to measure the 60-degree inner angles of the equilateral triangle and then entered those angles into their programs. That did not work as the robot kept rolling in different directions. The Sphero robots are designed to use a geocentric reference system rather than an egocentric system. Geocentric systems use a global reference frame based on the external environment like the GPS or compass. Sphero robots are designed to move forward (0 degrees), move right (90 degrees), etc. using an internally designed compass system. The students intuitively use their bodies to differentiate between geocentric and egocentric systems. In an egocentric system, the robot's perception and navigation are based on its own local frame of reference. The students without an understanding of these systems relied on their bodies to align, map, and ground their experience with the robot's coordinate system.

Connecting learning experiences with previous knowledge: This vignette provides evidence that the learners used embodiment to connect abstract concepts to their previous knowledge. For the Sphero to navigate an equilateral triangle

(mathematically 60 degrees for all three sides), the Sphero robot will not navigate at 60-degree inner angles. To navigate the first side, the robot will choose the smallest angle between 30, 60 and 90 degrees. It will roll 30 degrees and then add 120 degrees on each right turn to make 150 degrees and 270 degrees respectively. So the angles for an equilateral triangle Sphero will roll could be 30 degrees, 150 degrees, or 270 degrees. The students demonstrated their prior knowledge of equilateral triangles by entering 60 degrees for each angle. Learners did not immediately understand the geocentric implication of using the virtual protractor to find angles. They relied on their prior knowledge of equilateral triangle angles (60-degree angles). This was a challenging task for the students, and they relied heavily on their bodies to figure out the directions of the angles. The students used embodied representations as a tool to ground new perspectives on understanding angles in the context of programming robots.

This vignette provides evidence that embodied representations can allow students to engage with the CT concept of abstraction by using their bodies to support their reasoning. By enacting the robot's movement through his own body, the student was able to reason about spatial relationships and angles involved in the task, which in turn helped him adjust his prior understanding of equilateral triangles in geocentric contexts and improve his inputs to achieve the desired outcome.

Interconnectedness of Embodied Representations and other forms of Representations

Embodied gestures were interconnected with other forms of representations such as concrete representations (the triangle) and symbolic representations to ground the learner’s understanding of angles. As shown in the analysis of the vignette above,

the learners did not only engage in embodied forms of representations. They relied on an interconnected system of representations to build on their understanding of how a geocentric system works to make decisions about angles on an equilateral triangle. Here are a few examples of how other forms of representations supported the learners' engagement in embodied practice:

Concrete Representation: The taped triangle on the floor invited students to "walk" the path thus bringing embodied resources to bear on the task. This provided a clear, visible reference the student could embody as they programmed the robot to follow the equilateral triangle path. Students were also seen using their fingers to map out the angles inside the triangle (Figure 53). The students used their fingers to draw arcs around the inner edges of the triangles. This action points to the idea that the students were drawing on their prior knowledge of angles to support their understanding of angles.



Figure 53: Students move their fingers on the inner edges of the triangle mapping out the angles.

Symbolic Representation: The virtual protractor on the iPad served as a symbolic representation of angles that supported the students to visualize angles in ways that they could abstract using their bodies and make changes to angles and directions on their programs. Embodied gestures interconnected with other representations such as concrete representations (the triangle) and symbolic representations to ground the learner's understanding of the angles of an equilateral triangle thus overcoming the complexity of understanding the geocentric system of the Sphero robot.

Discussions on Embodied Representations

This vignette provides evidence that understanding the differences between the Sphero robot's internal coordinate system and the geometric properties of an equilateral triangle was key to programming the robot to navigate the triangle. Learners' lack of prior experience in robotics systems and knowledge of geometry compelled the students to lean heavily on embodied practices to support them navigate tasks. The student's embodiment of the triangle by walking on the triangle's path, helped him link the geometric concept to his sensory experiences. The student's use of embodiment in this context demonstrates the important role that physical actions and experiences play in learning and problem-solving. By connecting abstract concepts to tangible actions, students can develop an intuitive understanding of complex tasks, leading to a strong grounding in CT concepts through abstractions.

In this example, I have provided evidence of how learners interacted with angles in a context different from pen and paper contexts. They were able to realize that measuring the inner angle (which is the case in pen and paper activities) and programming the angle for a robot to navigate would not arrive at the same outcomes. Such activities provide opportunities for teachers to engage learners to think about angles in practical ways that can deepen their perspectives of angles in contexts different from 2-dimension pen and paper contexts.

By combining embodied representations, concrete representations, language representations, and mathematical knowledge, the students were able to abstract and develop an understanding of angles in contexts where the robot is controlled using a global reference frame based on the external environment like the GPS or compass. Spatial reasoning played a critical role in this activity as the students needed to understand and manipulate the spatial relationships between the robot, its surroundings, and the target locations indicated by the global reference frame.

Interconnectedness of Representations

In this section, I use a vignette to focus on the power of interconnectedness of representations in CT tasks. The interconnectedness of representations is important to highlight because it brings out the dynamic relationship between various representations and their role in problem-solving. As demonstrated in the previous sections, representations in CT tasks play an important role in abstracting and expressing CT knowledge. The interconnectedness of representations extends to the synergy between abstract and concrete forms of representations. Abstract concepts

can be translated into concrete, tangible models that can help learners understand and visualize concepts. For example, a complex algorithm statement can be represented visually through programmable blocks with vocabulary to translate commands, making it more accessible for young learners without programming syntax to understand. In the following section, I use a vignette to investigate how one student used representations in an interconnected fashion in a carousel repair activity on the Sphero. Math curriculum to describe a carousel animal repair task.

Description of Carousel Repair Activity

Carousel repair Activity: The carousel repair activity as presented on the Sphero.Math activity requires students to use angles and a function to help Sphero fix a broken carousel. This activity is lesson number 9 on the Sphero.Math curriculum. Teachers designed and handed students a carousel repair activity sheet to mimic a carousel. As shown in the figure below (figure 54). Students were required to program Sphero robots, visit chosen animals, and at the same time command Sphero to move through different angles between 0 - 360 degrees in 45-degree increments.

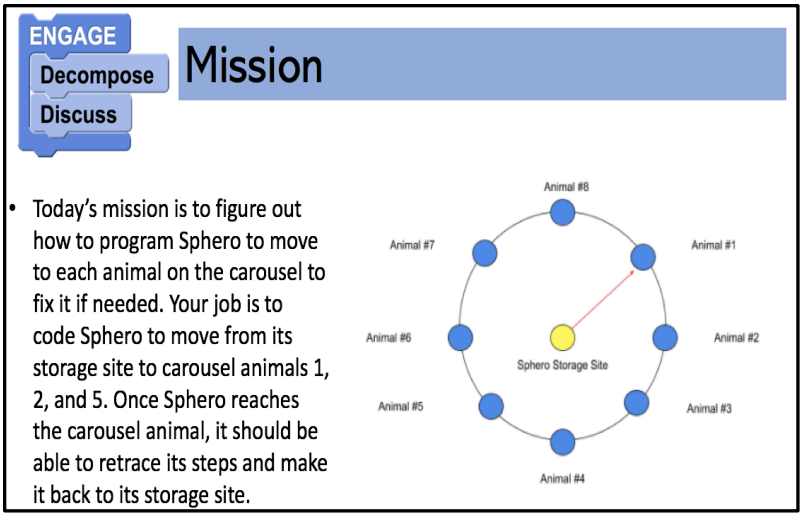


Figure 54: Carousel repair activity on Sphero.Math curriculum.

The carousel repair activity involved using a large sheet of paper and stickers to model a carousel (figure 55) and having students program the Sphero robot to move through angles to complete the activity. The vignette presented below demonstrates a student's conversation and interactions during a carousel repair activity with an interviewer as he tries to program the Sphero robot to mimic a carousel repair activity. This vignette captures the full activity during a lesson that required students to program the Sphero to move to each animal on the carousel to fix it as needed.

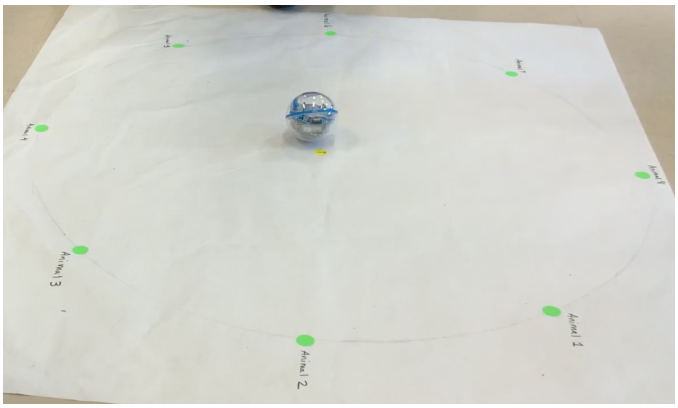


Figure 55: Carousel repair activity sheet.

Vignette Script for Interconnected Representations

Interviewer: Can you tell me how this works? (referring to the whole carousel activity, the code on the iPad, and the carousel model sheet)

Student 1: When you create a function block, you get a define block, and then you have to define the blocks that already exist what each block does. So normally, without the function block it we'd have to do like blocks of code, err like these same code blocks, we won't be using these but we would like errr oh, aim (moving his hand in a forward motion to indicate straight ahead - which aligns with the aim position, and then moving it rapidly to the right to indicate a turn, and then with a finger gesture indicates come back to the start position) roll at 50 degrees, blink, come back. We have to do that with multiple blocks so this (pointing to the function block on his computer screen) can be programmed with just one block. The define block. The define block defines what the block is supposed to do, so it's pretty simple and I have to adjust the numbers.

Student 1: It's pretty simple and I have to adjust the numbers (inaudible)

Interviewer: so how does it know when you say visit, when you say visit animal 7 how do you know, what makes it visit animals in the position7?

Student 1:errmm, this is the whole thing, it says 45 degrees, then times, the number so it will know the number, what I put, if it takes on this part only this block (scrolling to a different function on the screen) or this one, up here oh we've done that one now this one, this will switch to a different number

Interviewer: hmmm

Student 1: That's what I don't know, that's what I want to see okay fantastic, let's do it then (student then grabs the Sphero from the floor and then positions it on the activity.

Interviewer: Are you aiming it, yeah, make sure it is (inaudible)

Student 1: So, eight is supposed to be zero (stretching his right hand in front of him to make a straight line and align with the position of animal 8 on the activity

Student 1: That's technically what told it to do (when the robot visits the animals on his code function)

Interviewer: To go to five and to go to one?

Interviewer: oop goes to 7 ..wow(when the robot makes some moves in the wrong direction

Student 1 picks up the robot and says" I think (inaudible)

Interviewer: Can I show you something and can I ask you a question? What we just did is we added the delay block twice, what would happen, so what would happen if we did this/(Adding a loop block)

Student 1: So, the same effect the same effect would happen, the visit animal is programmed to do what it is supposed to do.

Analysis of Interconnectedness Vignette

The student begins by explaining how the activity works by saying *"When you create a function block, you get a define block, and then you have to define the blocks that already exist and what each block does".* When the interviewer asks the student *"So how does it know when you say visit, when you say visit animal 7 how do you know, what makes it visit animals in the position 7?"* The student provides explanations on the robots' behavior on the carousel. He explains how 45-degree increments are obtained each time an animal is selected saying, "*errmm, this is the whole thing, it says 45 degrees, then times, the number so it will know the number.* In

reading the function block, the student engages in symbolic and language representations by visualizing and reading the function block. The student also articulates his mathematical knowledge and understanding of the robot's responses to input.

The student continues in his explanations; "*so eight is supposed to be zero" (stretching his right hand in front of him to make a straight line and align with the position of animal 8 on the activity)* to explain how the robot visits animals saying*, "that's technically what told it to do" (when the robot visits the animals on his code function).* The student uses language and embodied representation to explain angles on the carousel, metaphorically connecting the position of animal eight to zero and embodying zero with a stretch of his arm to represent a straight line. The student starts by explaining using language representations, embodied representations, and symbolic representations to describe how to program the function block which is a symbolic representation (figure 58).

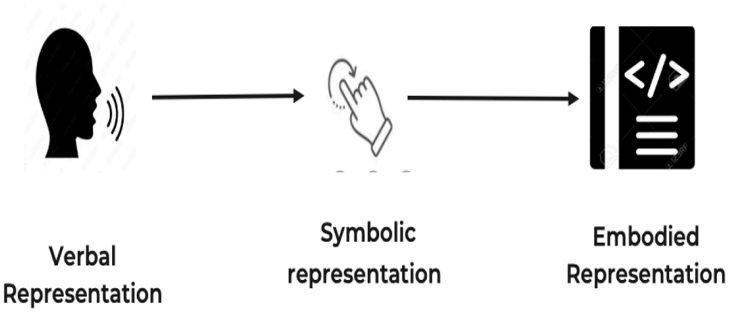
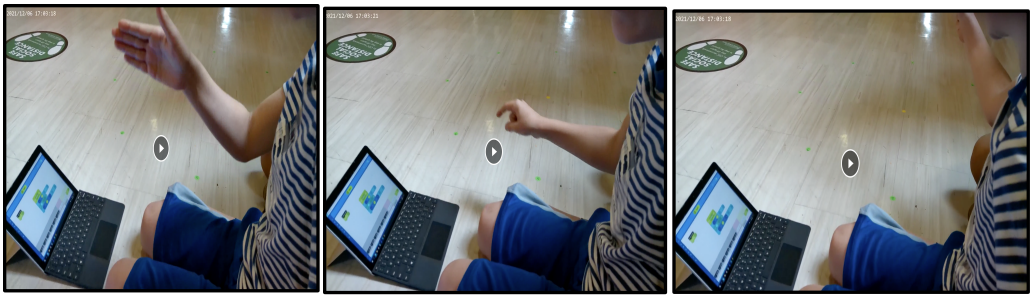


Figure 56: Interconnected meaningful connections of representations from verbal to symbolic and embodied representations a student navigates to problem solve.

After expressing his understanding of the function block, the student engages in a series of representations to express his understanding of the task; The student uses language representations to explain how the function block works saying; *"So normally, without the function block it we'd have to do like blocks of code, err like these same code blocks, we won't be using these but we would like errr oh, aim (moving his hand in a forward motion* in an embodied representation figure 57a) *to indicate straight ahead - which aligns with the aim position, and then moving it rapidly to the right to indicate a turn, and then with a finger gesture indicates come back to the start position figure) roll at 50 degrees, blink, come back".* In this explanation, the student engages in a series of embodied gestures to support his explanations of how to create and program the function block.

I argue that it takes some level of representational fluency, depending on the intensity of the simulated actions triggered by the activity to engage in these simultaneous interconnected representations (figure 57b). This student shows high levels of proficiency in algorithm design abstracting concepts like functions to translate between virtual symbolic representations (the code blocks), language, and embodied representations. There is evidence from the vignette that the learner engaged in gesturing because of the multimodal visualizations provided by the physical movements or actions of the robot combined with the language explanations he had to provide resulting in the generation of multiple forms of representations.



(a) (b) (c)

Figure 57: Student embodies the code and a physical representation of the robot (the "aim" light) as he explains the function of the blocks in his code.

(a). The student explains the "aim" position using his hand stretched in front of him to indicate a forward movement. (b). The student indicates a change of direction command in the code by moving his arm to the right. (c). The student indicates end in the code or stops using his finger to point down to indicate stop.

The interviewer asks the student to consider the impact of adding a delay block and a loop block; all symbolic representations used to indicate stops and repeating tasks; *"Can I show you something and can I ask you a question? What we just did is we added the delay block twice, what would happen, so what would happen if we did this (Adding a loop block), student 1 responds "So the same effect would happen, the visit animal is programmed to do what it is supposed to do".* The student demonstrates an understanding of symbolic representations (loop block and other blocks) put together to build the algorithm design (figure 58).

The use of the loop block means the learner recognizes a pattern recognition and uses loops to design the algorithm, thereby engaging in CT components of pattern recognition and algorithm design. Here are the step-by-step explanations of the function code as described by the student; *"We have to do that with multiple blocks*

so this (pointing to the function block on his computer screen) can be programmed with just one block (figure 58). The define block. The define block defines what the block is supposed to do, so it's pretty simple and I have to adjust the numbers. The student uses language representations to decompose the steps necessary to create the function code (figure 58) that can be used to automate the activity. The student's use of language and symbolic representations to decompose and abstract the activity and build the algorithm provides evidence of the supportiveness representations can lend to the enactment of CT practice.

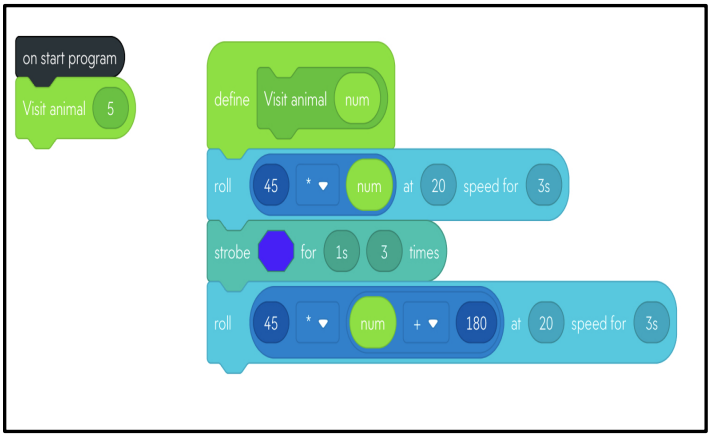


Figure 58: Function block, define block with its child codes that carry the instructions the function will execute.

Discussion

In this vignette, the student was able to use a simulation of a carousel to program the Sphero robot, to visit selected animals and at the same time program Sphero to move through different angles between 0 - 360 degrees in 45-degree

increments. This activity demonstrated how the student could rely on different forms of representations, concrete provided by the carousel, symbolic (programmable blocks), embodied (gestures), and language representations to communicate his understanding of how to build a program that involved the use of functions to control the Sphero robot to move around points on a circle in 45-degree increments.

The student engaged in all the CT components of Decomposition, pattern recognition, abstraction, and algorithm design. Additionally, the student used representations to express two important concepts in mathematics and computer science (CS) that are usually considered abstract; angles and functions. The vignette showed evidence that the student's understanding of complex concepts can be observed and understood by the fluency with which he navigated the representations. In CS, functions are usually modular and reusable and combined to create larger programs, making them not very easy to navigate and understand especially by young learners. It is therefore usually not obvious to immediately tell parts of a function once it's integrated into larger programs making them appear abstract conceptually. In this activity, the learner was able to rely on representations to describe all the parts of the function concerning the role the functions play in programming the robot.

In mathematics, angles are considered abstract and concrete at the same time. On one hand, angles can be visualized and measured using concrete tools such as protractors or using hands or other body parts to demonstrate different angles that represent directions or directions the robot will move. The student navigated the concept of angles using embodied representations, relating to other mathematical relationships to calculate and define the directions and angles the robot would move.

All these abstractions used to simplify angles were situated within the interconnected use of representations, through explicit explanations, and visual cues that connected information from one representation to the other, consequently supporting the student's understanding of complex concepts.

The GSA framework argues that gestures are a result of representations involving highly simulated actions (Hostetter & Alibali, 2010). Action simulations refer to mental processes in which individuals mentally simulate or envision actions, movements, or behaviors without physically performing them (Hostetter & Alibali, 2008). I argue that these simulations as in thinking often involve imagining and mentally rehearsing or arranging ideas, actions, or sequences of actions in the mind, which can come as a prelude to problem-solving or physical hands-on tasks.

In this study, interconnected representations connect two frameworks, the Lesh Transition Model and the GSA framework to provide evidence that in highly multimodal environments such as the CT robotics programming environment, highly activated simulations of actions can be omnipresent leading to the generation of different forms of representations. In other words, the nature and fluency of representations depend on the intensity of activation of simulations. In exploring representations and the interconnectedness of representations, I provide evidence of the role representations play in the acquisition of CT skills for young learners through scenarios that can be generalized in complex decision-making contexts where precision skills are essential.

Conclusion

In conclusion, my findings have explored the use of various forms of representations - concrete, embodied, graphic, language, and symbolic, in CT robotics tasks to answer the question; *What kinds of representations and expressions do young learners engage to interact with Sphero robots in a mathematics classroom and how do these representations contribute to successful problem solving in CT tasks?* Through an extensive analysis of the literature and practical examples through vignettes, I have demonstrated the role of five forms of representations, concrete, graphic, embodied, language, and symbolic representations in supporting CT practice and skill acquisition. My findings point to evidence that representations play an important role in supporting young learners to acquire CT skills such as Pattern recognition, algorithm design, problem decomposition and abstraction (PRADA). I provided evidence to answer the first research question, demonstrating that representation, whether created by learners, integrated into the programming interface, or inherent in the activity significantly contributes to supporting learners navigate CT tasks.

Furthermore, my findings also emphasize that representations when enacted in complex multimodal environments can rarely be expressed as standalone entities. They tend to co-exist as part of larger, interconnected systems that are either unique to an individual (personal and idiosyncratic) or shared by a group or culture. The interconnected use of representations plays a vital role in promoting students' development of CT practice/knowledge. These various forms of representation work together to ground and support students' problem-solving efforts, enabling them to

develop a more holistic understanding of either a complex concept or problem and its potential solutions.

However, students may sometimes encounter challenges in understanding the relationships between different representations or translating between various forms of representations effectively. This challenge was evidenced in the vignette featuring embodied representations where the geocentric nature of the robot's compass system challenged the student's understanding of angles causing him to abstract with his body and the virtual protractor. Despite the learner exploring a variety of representations, meaningful interconnection among them posed significant challenges. Therefore, it becomes imperative for educators to offer additional scaffolding and support to assist students in developing the essential skills required to navigate these interconnected systems successfully.

Finally, I have connected two frameworks, the Lesh Transition Model and the GSA framework to provide evidence that CT robotics programming environments can lead to the generation of different forms of representations (concrete, graphic, language, symbolic, and embodied) that can significantly enhance students' problem-solving abilities. By recognizing and embracing activities that use representational knowledge, educators can create a rich and supportive learning environment that fosters the development of CT skills and communication skills for young learners.

These findings contribute to a deeper understanding of the role that representations play in CT learning, especially in environments where intelligent agents such as robots are introduced. My work highlights the importance of nurturing

a diverse and interconnected representational approach to problem-solving in the context of CT and robotics tasks. Students in the process of translating across representations during these tasks, translate across complex concepts in ways that can enhance their problem-solving skills and overall understanding of CT tasks.

Chapter 6: Discussion and Conclusion

There has been a lot of research and advocacy devoted to the integration of CT within classrooms, either through plugged or unplugged activities. This study's effort aims to support research around the ever-growing presence of computational devices that play important roles in our daily lives. Initiatives led by organizations such as CSTA, code.org, CSforAll, Google's CS education, and several other initiatives have been actively involved in developing comprehensive curriculum guidelines and garnering political support at both the federal and state levels to promote the integration of CT in K-12 education. Likewise, there has been considerable research to understand how best to support learners in K-12 settings to acquire CT skills (National Research Council, 2010; Shute et al., 2017; Wing, 2006; Weintrop et al., 2021; Clements, 1999; Barr & Stephenson, 2011; Sengupta et al., 2013; Weintrop et al., 2016; Wilensky et al., 2014, Bih et al., 2020).

The idea that young children benefit most from hands-on learning with tangible objects can be traced back to the theories of Piaget (1970), Bruner (1966), and Montessori (1917). Piaget's work has led educators to embrace the idea that the cognitive processes of elementary school children are primarily concrete in nature, suggesting that they learn most effectively when presented with physical, tangible objects. It is assumed that children of this age group may struggle with mental operations involving abstract symbols. Other scholars have emphasized that the thinking patterns of elementary-school children tend to focus on concrete aspects of learning (Bruner, 1966). Bruner argued the use of concrete objects in teaching by

exposing them to a variety of concrete objects can help *shift children's focus from object's individual perceptual features to its abstract properties (p.65).* Alibali and Nathan (2012), Macedonia and von Kriegstein (2012), and Roth and Lawless (2002) has explored the cognitive benefits of gestures and movements.

It is in this perspective, that this study, titled; "Computational Thinking in Early Grade Classrooms: How Young Learners Interact with Physical Devices to ground their understanding of Computational Thinking", I observed how learners interacted with activities on the Sphero.Math curriculum to program the Sphero robot in mathematics classrooms. Using the Sphero.Math curriculum, developed through a collaborative effort between researchers and DCPS partners, I collected and analyzed data from fourth-grade mathematics classrooms and presented findings of how learners develop CT skills when they program Sphero robots to navigate tasks. My focus was specifically directed toward the kinds of representations learners used, considering their *perspectives (understanding)* of the environment, and the *practices* they engaged in to accomplish given tasks.

This study takes its roots from a larger body of literature that not only supports the use of tangible objects to make problems more accessible for young learners, but also explores how these learners can construct knowledge using the tools in the environment and form representations within the activities in ways that can help their understanding and problem-solving abilities (Piaget, 1968), enactivism (Varela et al., 1991), phenomenology (Merleau-Ponty, 1945/2005), pragmatism (Dewey, 1944), pedagogy literature (Montessori, 1967; Skemp, 1976; Rousseau, 1979; Freudenthal, 1983; Froebel, 2005), other more recent studies (e.g., Papert, 1980; Glasersfeld, 1987;

Wilensky, 1991; Pirie and Kieren, 1992; Steffe and Kieren, 1994; de Freitas and Sinclair, 2014; Sengupta et al., 2013; Resnick et al., 2005; Kafai & Burke, 2013; Holbert et al., n.d.). This study has contributed to support literature that has argued computers have been used as a valuable tool to empower young learners' engagement and hands-on experiential learning. The introduction of computational tools in learning environments has remained a central objective for constructionists, to honor children as creators, and promote their active engagement in constructing meaning for themselves through hands-on experiences, by democratizing access to powerful and creative tools available in the world.

This study has as its objective to integrate constructionist and programming activities into early-grade classrooms. To achieve this, I Investigated the use of developmentally appropriate tools and interventions to meet the cognitive demands of learners engaging with CT concepts in an interdisciplinary context (Papert, 1980; Resnick, 2012; Stoeckelmayr et al., 2011; Bers, 2010; Bers et al, 2014; Kazakoff, Sullivan, & Bers, 2013; Alimisis & Kynigos, 2009; Benetti, 2012; Bredenfeld, Hofmann, & Steinbauer, 2010; Johnson, 2003; Boticki, Pivalica, & Seow, 2018). Using qualitative analysis methods, I observed the integration of CT in mathematics elementary classrooms, using programmable Sphero robots to investigate how representations and the design of the Sphero robot and its programming environment can support young children learn to "think computationally".

This was achieved by examining how learners' conceptual understanding of CT aligns with four fundamental CT concepts: Pattern Recognition, Abstraction, Decomposition, and Algorithms (PRADA), as defined in the Sphero.Math curriculum. To support the

introduction of computational tools for young learners to support CT development, I used two dimensions to analyze the data around this study; (1) The design of the CT and educational technology tools used in the learning environment (Ching et al, 2018) and (2) The kinds of representations learners used to interact with the Sphero robot.

How Young Learners Interact with the Sphero Robot to Ground their Understanding of Computational Thinking

In this dissertation, I present findings showing how representations and the design of computational tools can support children's interactions with CT practice. My findings revealed that representations play an important role in supporting young learners to engage in CT practices such as Pattern recognition, algorithm design, problem decomposition, and abstraction (PRADA). Representations such as (1) concrete manipulatives, (2) language, (3) graphic, (4) symbolic, and (5) embodied

representations provided scaffolds for learners to gain CT skills by supporting learners to iterate, test, debug, abstract, modularize, and reuse code.

Additionally, design features of the Sphero robot and its programming environment supported learners to gain CT knowledge. Features such as (1) programmable LEDs provided opportunities for learners to break down tasks and create opportunities to organize and structure components to get visual feedback that helped them recognize patterns. (2) Taillight ("aim") for LED/geocentric orientation, provided visual cues, such as the "aim" light on the robot, facilitated the involvement of embodied practices, empowering students to establish sensorimotor references. (3) Sphero's virtual protractor supported students through the CT component of abstraction to address the geocentric aspects of the Sphero robot. (4) block-based environment/language, that involves the use of shapes and colors as effective visual aids and abstraction tools, to support learners to construct algorithms.

Chapter 2 of this study reviewed previous studies on CT and representations, and explored how integrating technological tools into classrooms promotes experiential learning through a hands-on constructionist approach to learning (Papert, 1980). These constructionist approaches (programming computers), it has been argued, can develop and support children's cognitive abilities (Papert, 1980; Wing, 2006). Additionally, chapter 2 explores the role of representations in CT-enhanced environments, by looking at frameworks that support representations in learning environments.

Chapter 3 presented the methods around which this study was designed, focusing on the methods used to design the study. I collected video images from focal

groups of students in their respective classrooms selected by their teachers and conducted a qualitative study after having observed and analyzed recorded videos of participants using interaction analysis (IA) as the main approach to analyzing data. I use the Lesh Transition Model (Lesh and Doerr, 2003) and GSA by Hostetter and Alibali, (2010) as frameworks through which I analyzed representations. I use a framework on CT concepts and practices defined by the Sphero.Math project guidelines (see table 2.4) to analyze CT practices such as Pattern recognition, algorithm design, problem decomposition, and abstraction (PRADA).

Chapter 4 of this study explores the features of the Sphero robot and its accompanying programming environment, and how these features support students' interactions with CT knowledge. To explore these characteristics, Chapter 4 answered the question: *What features of the Sphero robot and its accompanying programming environments facilitate students' understanding of CT knowledge?*

In chapter 5, I analyzed data collected from focal groups and highlighted relevant topics to explore around how representations can support student interactions in CT-enhanced learning environments. Analysis provided evidence to answer my first research question; *What kinds of representations and expressions do young learners engage to interact with Sphero robots in a mathematics classroom and how do these representations contribute to successful problem-solving in CT tasks?*

In this final chapter of this study, I provide a summary of my findings from two chapters. First, I share discussions about my findings for Chapter 4 on how I explored features and design choices of the Sphero robot and its accompanying programming environment, that supported CT learning for young learners. I present the results of

my analysis in chapter 5, which focused on how I applied the Lesh Transition Model (Lesh & Doerr, 2003) to investigate how learners transitioned between four different forms of representations together with the Gestures as Simulated Action (GSA) framework by Hostetter and Alibali, (2008) to understand representational gestures (embodied cognition) produced when actions are simulated as part of thinking and speaking.

My first research question asked: *What features of the Sphero robot and its accompanying programming environments facilitate students' understanding of CT knowledge?* I found that certain design aspects of the Sphero robot and its programming environment support students' understanding of CT knowledge by simplifying the complexity associated with navigating the programming environment and designing algorithms. I provide vignettes with evidence that supports the idea that certain design features of the Sphero robot can facilitate young learners' interactions with CT knowledge. For example, the Sphero robot has LEDs that can be programmed to display different colors and patterns. In the vignettes presented, learners used LEDs to visualize and understand events about the activity.

In Sphero.Math activity #8 titled "Monorail Stops," learners programmed the Sphero robot to travel from the "Start" block to the "Stop" block, making stops at each "bus stop" and flashing its lights from yellow to green. I argue that programmable LEDs can be used to provide visual feedback for the learners. Design principles around visualizations and their potential to support understanding and hypothesis formation have been argued by Resnick and colleagues, (2005). This idea of using LEDs to offer visual cues that can delineate distinct stages of the sequence

through decomposition can help students identify patterns within the activity. These actions fall within the CT components of decomposition and pattern recognition in the PRADA framework.

In addition to the LEDs built into the robot, I also presented vignettes where students used parts of their bodies as a cognitive support tool to help them align to the Sphero "aim" light or desired heading. Vignettes that involved students engaging with sensory motors such as their feet, fingers, and eyes embodied practices to align with the robot's aim lights to find orientations between the robot and the activity were presented. I analyzed embodied practices through the lens of literature on GSA (Hostetter & Alibali, 2010; Nathan et al., 2017) and embodied cognition theories (Barsalou, 1999; Glenberg, 1997; Lakoff, 2012), that argue abstractions are metaphorically grounded in embodied and situated knowledge.

I present the "aim" light as a symbol used to indicate the heading or the direction the robot will travel. I argued that symbols serve as representations or abstractions that simplify complex ideas. I further argued that when young learners engage in embodied practices in CT environments, embodiment supports them to connect with visual cues, expressed through a sense of their body's position and movement, thereby enriching the learning experience (Weisberg & Newcombe, 2017). Additionally, embodied perspectives created analogies (direction of feet being analogous to the heading) that align abstractions with the empirical system in ways that can support learning (Wilson, 2002). One of such analogies was presented in another vignette the Sphero app's visual representation of the protractor served as a support tool for students to navigate angles. The visual aspect of the protractor

provided scaffolds for students to form images and representations of the angles they encountered in the activity, thus supporting their ability to visualize and abstract the complex concept of angles using their bodies.

One requirement for designing CT learning tools/environments for young learners is to attain the goal of making them age-appropriate or developmentally appropriate. This involves making it easy for the tools to support young learners learn difficult concepts in experiential ways (Brennan & Resnick, 2012, p.3; Ching et al., 2018; E. Kazakoff & Bers, 2012, Mioduser & Levy, 2010). This goal was attained in the Sphero programming environment design by organizing blocks into categories with distinct shapes, block-based programming environments provide a visually organized and structured environment where students can memorize and keep track of block categories by shape and color. In the vignette I presented in chapter 4 to support the design of the programming environment, students were tasked with modifying a program to announce the distance Sphero rolls on collision with the wall. The students identified the distance block using the block's color.

This way of identifying blocks by abstracting and generalizing blocks using color provides evidence that the choice of categorizing blocks by color provides visual support for students to navigate through the blocks categories. By organizing blocks into categories with distinct shapes, block-based programming environments were able to provide a visually organized and structured environment where students can memorize and keep track of blocks categories by shape and color. Learners used CT component of abstraction to help them manage the complexity of the

programming environment, making it more manageable and understandable for them to navigate.

Evidence suggests that designers of language toolkits often encounter challenges in striking a balance when it comes to designing tools suitable for various age groups (Resnick et al., 2005). The more user-friendly toolkits become, the fewer functions they tend to offer. However, even when they possess the same level of capability, there exist simpler and more complex ways to present these functions (Resnick et al., 2005). I provide evidence that shapes and colors of the blocks can be used to categorize and recognize blocks to use within any category. This study contributes to existing literature on block programming environments by highlighting their capacity to make it clear to users what actions are feasible (Weintrop, 2019; Lin & Weintrop, 2021).

In the Sphero.Math activities, the choice of programming the robots using block programming language enabled the students to express themselves and develop their programs. While examining the design of the tools used in the Sphero.Math activities, there was a prominent demonstration of Abstraction as a key component of computational thinking (CT)

I argue in support of Wing's definition of CT, that the key characteristic of abstraction is the concealment of complexity. Categorization into colors and shapes were key design features that supported learners to author programs without having to deal with the constrains of understanding the underlying complexity around the programming language. This finding underscores the importance for educators and designers to recognize the dual nature of CT abstractions. While learners may create

and use certain abstractions themselves, others are inherently embedded within the design of tools or environments. This suggests that educators and designers should strive to create learning environments and tools that not only encourage learners to develop their own abstractions but also provide support by incorporating pre-existing abstractions, thus lowering the barrier for learners to interact with CT practices and concepts.

My second research question sought to understand: *What kinds of representations and expressions do young learners engage to interact with Sphero robots in a mathematics classroom and how do these representations contribute to successful problem-solving in CT tasks?* I found that representations play an important role in supporting young learners to engage in CT practices such as Pattern recognition, algorithm design, problem decomposition, and abstraction (PRADA). My findings showed evidence of students using concrete manipulatives, language, graphic, symbolic, and embodied representations to gain CT skills through iterating, testing, debugging, abstracting, modularizing, and reusing code. The use of representations played an important role in providing different dimensions of CT perspectives to learners by providing them pathways to express their ideas, ask questions, and think creatively as they problem-solve. Learners used the following five forms of representations to engage in different dimensions of CT practice:

a. *Concrete manipulatives as forms of representations*: In the vignette used to analyze concrete manipulatives as representations, I position the monorail path, designed into chapter 8 of the Sphero.Math curriculum as a concrete manipulative that helped

learners develop CT skills. The monorail path is designed in the form of a maze. This design supported students to decompose tasks into smaller manageable steps providing opportunities for learners to logically organize parts that came first. This organization enabled students to find patterns and overcome complexities by abstracting either through intuitive sense or supported by other forms of representation learners could afford. Learners in this activity engaged in CT components of decomposition, pattern recognition, abstraction, and algorithm design. The concrete monorail path provided the required support the students needed to translate information from the physical activity space to create algorithms to virtually control the robot.

The monorail path vignette example reinforces the idea that computer science can incorporate engineering thinking to create systems capable of interacting with the physical world (Wing, 2006). The constraints of getting information from the physical space to the virtual space, forced students to think computationally. The computational concepts the learners used to collect information from a physical monorail path to a virtual space required the learners to "think computationally". Segupta et al. (2013) argued that CT goes beyond the term "thinking". Segupta further argues CT is grounded in disciplinary and representational contexts that can portray abstractions in use. In the context of this study, I draw on mathematics literature that argues manipulatives can enable students to connect abstract mathematical concepts to real objects (Uttal et al., 1997). I argue that the monorail path as a concrete manipulative, supported students to use representations to abstract

complex concepts to simpler concepts consequently acquiring and expressing CT knowledge.

The design of the monorail path activity is also grounded in constructivist literature, that argues learners can develop scientific thinking by intuitively building upon and refining prior knowledge (Sengupta 2011; Dickes and Sengupta 2012; diSessa 1993; Hammer 1996; Smith et al. 1993; Sengupta and Wilensky 2009, 2011). This idea was demonstrated in the concrete manipulative representation example when students leveraged on naive conceptual understanding of decomposing the monorail path, by counting the squares and translating squares into time to measure distance.

Students relied on the concrete manipulative as a reference tool on which they relied to build a perceptual understanding and visual cues that helped them translate or create other forms of representations to help them problem-solve. For example, the concrete manipulative used in this activity was never used in isolation. It was intertwined with symbolic, embodied, and language representations, making the concrete manipulatives a translational tool in CT tasks. I argue that integrating concrete manipulatives into CT tasks can provide opportunities for learners to track their thinking by engaging in multiple forms of representations, initiate reasoning by creating other forms of representations, and obtain immediate feedback to problem-solve.

b. *Graphical Representations:* The monorail path, designed to mimic stops for a train in the Sphero.Math curriculum leverages on a design choice to make visual a

graphical representation of a path that uses symbols, squares, colors, and lines to guide the flow of control and the sequence of operations that the robot is programmed to follow. Visual imagery plays a very important role in mathematics teaching and learning (e.g., Lakeoff, 1987; Cuoco & Curcio, 2001; Romberg, Fennema, & Carpenter, 1993; Yerushalmy & Shternberg, 2001; Zimmermann & Cunningham, 1991). Humans use visualization, including prototype images, to enhance their thinking, processing, and reasoning (Jones, 2018). In this study, I use the monorail paths to represent graphic representations. Through aspects of concept encapsulation, I argue that a single square on the maze can store information related to variables such as speed time and total distance the robot must travel. In the selected vignette, students count the squares on the maze, attributing time values for each square thus encapsulating the concept of distance. The graphic representation provided evidence of the symbiotic relationship between mathematics and CT (Fofang et al., 2020). In this context, students used addition and iterative logic to express and apply CT practices effectively, mediated by the maze that afforded a visual guide for students to count squares, convert distances into time, and subsequently translate these calculations into a format a computer can execute.

The use of graphic representations supported the generalization of knowledge transfer by supporting the development of CT concepts such as decomposition, pattern recognition, and algorithm design. Learners were able to decompose and program segments of the maze, identify recurring patterns, and use loops to program. Graphic representations supported learners by providing visual aids that enabled them to understand patterns and relationships between variables. I provide evidence that

graphic representations can serve as the bridge between abstract CT knowledge and real-world applications of CT. By translating abstract concepts into a graphic form, learners can apply their understanding of CT from a 3D physical space to a 2D virtual space, mediated by representational translations that can support them create algorithms.

c. Language Representations: Learners used language throughout the activities while engaging with tasks. Language representations served as a transition tool supporting learners to translate from one representation to another. The language representations I used for this study included verbal expressions, discussions and, conversations related to the activities. The students and teacher I observed engaged with language representations by listening, speaking, reading, or writing about computational tasks and concepts. They used spoken language to verbalize thoughts, explain reasoning, ask questions, or discuss ideas (National Research Council, 2010).

The chosen vignette demonstrates how the teacher uses explanations and prompting to frame the activity for the students to work on. The activity is framed around the idea that many parks are big enough that it would require a lot of walking to get from one attraction to the other. Learners were encouraged to use different blocks the teacher described to program stops for the monorail. Some of the focal group students engaged in the activity by writing down the program steps in their notebooks and engaging in discussions with their peers.

Language representations were used as a first-step strategy to organize tasks. Language representations can sometimes be used to break down programming

languages into simpler understandable language for translating abstract CT concepts into executable algorithms. Jeannette Wing's definition of CT supports the idea that to think computationally involves the capacity to express ideas in computable form. This involves constructing representations using highly constrained computer languages, which, unlike natural languages, offer limited perspectives, abstractions, and articulation capabilities. I argue that novice programmers can benefit from navigating between imprecise natural language representations, sometimes expressed in informal discourse to highly constrained programming language. This process can involve using natural language to describe actions and processes or creating tangible signs like gestures that can aid semiotic transformations, eventually leading to the composition of formal and precise expressions for building computable code fragments.

This argument is supported by mathematics literature, that argues speakers systematically use other forms of representations like gestures when speaking because gestures are related to cognitive schemas. (Alibali, Bassok, Solomon, Syc, & Goldin-Meadow, 1999). In this study, I use evidence from language representations to suggest that, when young learners use robots to explore CT concepts, Wing's (2006) definition of CT, which emphasizes that "expressing ideas" in computational form, might rather represent the final stage of the CT skill acquisition process.

d. Symbolic representations: In this study, I refer to symbolic representations as the use of symbols or signs to represent concepts, objects, or relationships in a structured and meaningful way (Uttal et al., 1997). These symbols can be letters, numbers,

mathematical operators, or any other characters that have a specific meaning within a particular context. Symbolic representations are commonly used in mathematics, science, and programming languages, where precise and unambiguous communication is essential. In this study I analyzed vignettes to show how symbolic representations supported learners engage in CT practices and completed tasks.
The activity selected to showcase symbolic representations is an activity the students designed around an amusement park with five activities for visitors to navigate: a maze, the beach, a restaurant, a water park, and a roller coaster. The students used hashtags on the activity board to represent functions, dots to represent the robot's aim light and functions to program and separate each activity's code from the other. Students embedded color codes into the activities which they used to guide visitors using a color-coded manual they designed for users. I argued that symbolic representations can support learners engage in several dimensions of CT practice:
Abstraction to break down complex concepts such as functions in the virtual space to simplified symbols like hashtags on the physical activity.
Problem Decomposition: Learners used symbols to decompose the activity by using functions for each component allowing learners to focus on individual components.
Modularity: I argue that the student's choice of using functions to distinguish each activity on the amusement park is evidence that the students are designing the system as a collection of independent, decomposable components.

I argue that, in CT contexts, symbolic representations can provide a standardized and concise way for learners to communicate ideas within a community of practice. For example, block programming languages may incorporate a fusion of language

and symbolic representations (Bau et al., 2017). For example, block languages seamlessly combine linguistic and symbolic representations, visual cues provided by blocks' shapes and colors guide their assembly (Lin & Weintrop, 2021; Maloney et al., 2010). For example, in mathematics, standard symbols are used to communicate ideas and notions. This can be crucial in collaborative learning environments when sharing ideas or solutions with others. In consistency with Mathematics literature, I argue that symbolic representations in CT environments either standardized or created by the learners can help students connect and build links between individual symbols and their referents (Wearne & Hiebert, 1988, p. 372). Written mathematical symbols are usually presented before the manipulatives or referents, making manipulatives a bridge to help learners connect and master the abstract symbols otherwise they may never make the connections on their own. In this study, I argue the hashtags and functions served as symbols that were bridged by the robot, the amusement park activities and the program designed to control the robot. I argue that the use of symbolic representations in CT robotics programming activities constitute a form of CT practice (Abstraction) and learners' computational thinking can be observed through their successful use of representations as they translate between physical, and virtual spaces.

e. Embodied Representations: Students engaged in embodied practices as they engaged in Sphero.Math activities. Given the multimodal nature of these activities, referencing mathematics literature is especially beneficial to understand how learners use gestures and other embodied resources to navigate CT enhanced robotics

environments (Hostetter & Alibali, 2010; Nathan et al., 2017; Kirsh, 2013; White & Pea, 2011). In this study, I presented an analysis of a vignette in which students engaged in embodied practice as they programmed the Sphero robot to navigate a triangle. The vignette highlighted how the triangle tapped to the floor, served as a visual guide, and invited learners to "walk" the path of the triangle bringing embodied resources to the task. Analysis showed instances when learners embodied the triangle and angles by linking the geometric concepts to sensory experiences. Learners were able to realize that measuring the inner angle using a physical protractor (which is the case in pen and paper activities) and programming the angle for a robot to navigate would not arrive at the same outcomes.

The complexity of understanding geometric angles was abstracted using embodied practices. By connecting abstract concepts using tangible actions, students can develop an intuitive understanding of complex tasks, leading to grounding of CT concepts. In this study, I argue in favor of embodied theories of cognition which posit that the connection between action and perception, present in our physical interactions with the world, also manifests in our mental representations of the world (Barsalou, 1999; Glenberg, 1997). This means that, thoughts about visual-spatial images may sometimes include simulating actions connected to manipulations with the object. Learners engaging in embodied practices with the activities in this vignette is an indicator that they are fully immersed into the activity and are thinking about how to execute the activity (Wexler, Kosslyn, & Berthoz, 1998; Wohlschläger & Wohlschläger, 1998). In these states of thought, the GSA framework argues *that these highly activated action simulations are particularly likely to result in representational*

gestures. I argue that when learners walk the path of the triangle, use their arms to find directions the robot will roll and map out angles with their fingers, they are thinking about actions the robot will enact.

According to the GSA framework, neural simulations of actions, activate motor and premotor areas of the brain that can trigger representational gestures. I argue that activities such as the Sphero.Math activities that provide highly multimodal interactions for young learners are generative environments to observe kinesthetic qualities of students' motor actions that can lead to new forms of representations that can scaffold learning. In understanding embodied representations in CT tasks, this work contributes to literature that seeks to understand how embodiment can translate into CT practice thus generating new forms of learning in multimodal contexts.

Interconnectedness of Representations in CT Environments

I highlight in this study the idea that representations don't live on their own and do not always occur in isolation. Instead, they build on each other and interact with a variety of related representations that form a coherent structure to support problem-solving. This coherence in representations is built from action simulations that are a result of mental processes in which individuals mentally simulate or envision actions, movements, or behaviors without physically performing them (Hostetter & Alibali, 2007, 2008). I argue that such simulations same as in thinking often involve imagining and mentally rehearsing or arranging ideas, actions, or sequences of actions in the mind, which can come as a prelude to problem-solving in hands-on physical learning activities.

This study presents how interconnected representations connect two frameworks, the Lesh Transition Model, and the GSA framework to provide evidence that in highly multimodal environments such as CT robotics programming environments, highly activated simulations of actions can be omnipresent leading to the generation of different forms of representations. For example, in the vignette describing the interconnectedness of representations, the student engages in language representations to describe the functions (symbolic representations) using the carousel repair activity board as a visual reference (concrete representation) and engages in a series of gestures as he explains the task.

In consistency with the GSA framework (Hostetter & Alibali, 2008), there is evidence that this activity had highly activated action simulations by the abundance of gestures tied to other forms of representations. The GSA argues speakers gesture more when visualizing physical movements or actions without performing them physically than when describing visual images. There is some evidence to this claim as the student in the interconnectedness vignette and the teacher introducing the lesson in the language representation vignette gestured a lot more when describing motor information than in other vignettes where the students described visual images (Feyereisen & Havard, 1999; Hostetter & Alibali, 2007). I argue that the nature and fluency of representations depend on the intensity of activation of simulations, and this can be achieved through the design of the activities. This finding suggests significant implications for the educational community. Educators can leverage this insight by strategically designing activities that promote the activation of simulations, thereby enhancing the nature and fluency of representations among learners. By

understanding how the intensity of simulation activation influences representation proficiency, educators can tailor instructional strategies to optimize CT skill acquisition.

Variability in Representational Strategies Among Learners

In terms of programming the Sphero robot, learners used different strategies to navigate the activities to design programs. These strategies were sometimes linked to learners' levels of experience with programming. In chapter four, for example, the learner featured in the vignette demonstrating the carousel repair activity to illustrate the interconnectedness of representations and used a coherent pattern of gestures accompanied by language to effectively convey concepts. In contrast, the student in the embodied vignette (programming an equilateral triangle) faced challenges in elucidating concepts, used less language, and struggled to find angles and directions even after engaging with embodied resources. This disparity in strategy is explained by the GSA framework's assertion that gestures tend to emerge when actions are mentally simulated in support of thinking and speaking making speakers to be more inclined to use gestures when describing a representation that has been partially constructed through motor experiences compared to representations primarily shaped through visual experiences. This observation suggests that students with prior programming knowledge tend to rely less on embodied practice and are more inclined toward using advanced symbols and can create other abstract representations than relying on their bodies to understand concepts. The variety of approaches used by learners in this study demonstrates the dynamic nature of the learning process within CT programming environments. These insights contribute to our understanding of

how learners' representations can adapt and align with their developing proficiency and increasing familiarity with the activity and programming environment.

Implications

Implications of these Findings for Teachers

Given the continuous upsurge of computational tools in classrooms and learning environments and calls to broaden participation in CT within K -12 settings, teachers need to embrace professional development opportunities to enhance their own CT knowledge and instructional abilities. Although the objective of this study is not to establish the Sphero.Math projects as the definitive standard for CT education, this work can serve as a valuable resource for educators seeking innovative methods to incorporate CT into their teaching. For example, findings on language representations show how teachers can decompose and prompt students to think through the activity, especially through verbalizing parts of the codes using natural language in ways the learners can understand. This study can act as a reference guide, offering insights into the design and integration of CT-enhanced activities within interdisciplinary curricula contexts. These choices may involve the integration of CT into mathematics as demonstrated in this study, science, languages, and art, as highlighted by prior research (Blikstein and Wilensky 2009; Eisenberg 2002; Hambrusch et al. 2009; Rubin and Nemirovsky 1991; Sengupta et al. 2013; Settle et al. 2012, 2013; Weintrop et al., 2016. This study contributes to research that can support how to integrate CT in elementary grade classrooms using physical devices like programmable robots.

Research regarding the professional development of teachers in CT has mostly focused on their capacity to integrate CT principles into existing subjects (Walton et al., 2020). Findings in this work with regards to the role of representations can serve as a useful resource for experienced teachers willing to enhance their current pedagogic and curricula approaches by integrating more advanced CT practices into their skillsets. Successful implementation of CT can be determined by the learning experiences afforded by the learning materials. This study serves as a resource to answer questions around the design and choice of materials and their respective programming environments that can afford meaningful CT experiences, especially in early grade classrooms.

The push for equipping all students with CT skills, regardless of their backgrounds or prior experiences comes with certain implications. For teachers, this entails embracing inclusive teaching practices that cater for diverse learners. Teachers must consider students with varying levels of exposure to technology and adapt their teaching strategies to foster CT proficiency. This study provides insights on teaching practices and instructional abilities that can effectively guide elementary grade students to acquire CT skills. For example, in this study, I observed that learners who experienced traditional classroom instruction, with teachers providing constant guidance throughout the lesson, did not demonstrate the same levels of robust CT skills based on the forms of representations they engaged, in comparison to learners who were encouraged to explore and collaborate with their peers who generated coherent meaningful forms of representations that led to successful problem-solving.

This study can provide insights to teachers on the need to promote learning environments that encourage exploration and collaboration among learners. Traditional classroom instruction, characterized by constant guidance from teachers, may not effectively nurture robust CT skills in students. Educators should consider implementing pedagogical approaches that promote active engagement, and peer collaboration, which may sometimes lead to the generation of meaningful representations. By encouraging students to explore and collaborate, educators can facilitate the development of CT skills that are vital for successful problem-solving in various contexts.

These implications are profound, as not only does integrating CT into elementary grade disciplinary and curricula contexts empower students with valuable problem-solving and CT skills, but it also equips them to thrive in a technology-driven world, ultimately fostering a more equitable and inclusive educational landscape.

Implications of these Findings for Designers

This study has some implications for designers of programmable learning toolkits for young learners such as the Sphero robot and its programming environment. This study sheds light and can serve as a guide to support the critical role design elements such as the visual cues from the programmable LEDs, block-based programming environments can play in shaping students' understanding and engagement with CT concepts. For example, my findings support the importance of designing user-friendly interfaces, adding visual cues, and abstraction support within the programming environment to scaffold students' CT skills effectively. The “aim”

taillight on the Sphero designed the Sphero robot served as an abstraction tool learners could lean on using their bodies to align the robot to the heading or direction for the robot to roll. Likewise, findings around the integration of programmable LEDs that blink through the transparent shell of the Sphero robot proved to be a particularly useful addition to the range of events learners could use to visualize, decompose, and build algorithm flow. This study has been able to demonstrate that, in developing computational tools for young learners, designers should consider incorporating user-friendly interfaces that can support embodied practices, visual cues, and abstraction support within programming environments to effectively scaffold students' CT skills.

This study provides contributions for designers of programming environments to prioritize creating tools that support various forms of representations that can encourage students to explore, experiment, and problem-solve independently. For example, in Chapter 4, my findings suggest that activities that relied heavily on symbols led to the generation of other forms of abstractions to support learners navigate tasks. I provide evidence through a vignette on embodied representations of how a learner draws on embodied resources to connect his bodily movements to a virtual protractor and at the same time walk on a triangle path to find directions for the robot. This example suggests that there are gaps in the current interface in terms of how much scaffolding should be provided to support students in understanding the difference between directions and angles the robot will make. This challenge could also be evident due to the geocentric nature of the Sphero robot which sometimes challenges students to find alignment between internal and external angles on tasks with the robot’s geocentric compass.

The Sphero programming interface provides opportunities for learners to interact with the robot by lowering the bar for all age groups using tactile (drive and draw) functions, and block and text programming languages. The Sphero.Math curriculum activities specifically used block-based programs to control the robot, making the programming experience particularly generative for learners as shown in Chapter 5. For example, learners relied on the shapes and colors of the blocks to navigate the programming environment. This was particularly helpful for novice learners without any experience in programming. This study provides additional suggestions for designers to recognize the importance of incorporating intuitive and generalizable design elements, such as shapes and colors, within programming environments to facilitate navigation, especially for novice learners.

Through this study, I highlight the need for continued collaboration between educators and designers of educational technology tools to align with educational goals, and to ensure that CT learning experiences are accessible, engaging, and tailored to meet the needs of diverse learners. Designers should engage in partnerships and collaborative research efforts with researchers and educators to ensure that educational technology tools are informed by evidence-based practices and pedagogical insights, thereby enhancing their effectiveness in fostering learning outcomes, and meeting the diverse needs of learners. Thoughtful and intentional design choices have the potential to significantly enhance the integration of CT in K-12 education, empowering the next generation with essential computational skills for the future.

Implication of these Findings for Researchers

Representations as a pathway to understanding CT in early grade settings has not been extensively explored in the CT literature. This study contributes to the findings on the role of representations in the learning and understanding of CT in early-grade mathematics classrooms. Other than highlighting the important role representations can play in CT acquisition, this work also has demonstrated the mutually supportive nature of CT and mathematics in interdisciplinary contexts.

Just as this work can serve as a valuable resource for educators seeking innovative methods to incorporate CT into their teaching, it can also be considered a theoretical and practical contribution to our understanding of the nature of CT when viewed through the lens of representations in interdisciplinary contexts. Although numerous studies have explored the characteristics of CT in interdisciplinary contexts, few studies have focused on the role of representations in the development of CT skills especially in early-grade settings. This study uses two frameworks (the Lesh Transition Model and the GSA) to provide evidence that highly multimodal environments such as robotics programming environments can generate highly activated simulations of actions for young learners that can in turn trigger the generation and use of different forms of representations. These forms of representations can be used as evaluation tools to understand how young learners acquire CT skills. By highlighting the nature and interconnectedness of representations, I hope to contribute to the kinds of resources educational researchers can explore in their effort to evaluate CT skill acquisition in learning environments.

This study's innovative design of activities and curriculum offers a blueprint for researchers to understand how to effectively design and implement practical hands-on programming activities in curricula contexts for elementary-grade learners. It highlights the importance of creating technology-enhanced activities that align with the developmental stage of elementary-grade students while promoting CT and problem-solving. Moreover, the interdisciplinary approach of this study emphasizes the seamless integration of CT concepts into subjects like mathematics making it a valuable model for researchers aiming to bridge the gap between CT and traditional curricula. This study highlights the significance of introducing computational CT concepts at an early age, even in elementary school grades, challenging the conventional focus on CT within middle and high school education.

Researchers should recognize the expanding presence of computational tools in educational settings, which offers fertile ground for exploring the integration and impact of emerging technologies within learning environments. By acknowledging these opportunities, researchers can contribute to a deeper understanding of how CT can be effectively incorporated across various educational levels, ultimately informing the development of more inclusive and impactful educational practices.

Considerations for Implications: Insights for Education Across Grade Levels

Representations hold significant importance in mathematics education across all grade levels (Goldin, 1987, 1992a; Kaput, 1987, 1991). In early childhood education, learners are often introduced to numeracy skills using tangible tools, gradually transitioning to language, symbolic, graphic, and other representations as

they advance in grade level. As explored in Chapter 4, representations within CT environments take various forms, including concrete, graphic, language, symbolic, and embodied representations. While some representations may take continuous forms, such as concrete objects, others are discrete, such as spoken words or numerals. Symbols are commonly conveyed through tangible mediums like embodiment or concrete objects.

This study has provided evidence that meaningful representations are typically activated through action simulations, leading to interconnected representations that are part of structured thinking systems (Hostetter & Alibali, 2010). This characteristic of structured thinking shared by CT, mathematics, and science offers opportunities to enhance instructional approaches across disciplines aimed at developing knowledge among learners of all grade levels. By observing the evolution and progression of representational knowledge, educators can tailor instructional strategies to effectively support learners. For instance, Chapter 4 of this dissertation shows various abstraction techniques mediated by representations, facilitating the visualization of concepts and complex patterns within activities. This research has the potential to pave the way for educational research methodologies that integrate learning processes through visualization and analysis of representation properties, thereby enriching educational practices and outcomes across grade levels.

How these Findings Relate to Emerging Topics like AI, ML, and Data Science

The use of representations in CT robotics learning environments can be considered a gentle introduction to AI, ML, and data science concepts in education.

AI systems operate based on data representations that help them understand context and relationships within provided information. Representations could be textual, visual, or numerical. There is a symbiotic and intricate relationship between CT, representations, artificial intelligence (AI), machine learning (ML), and data science. CT, as a problem-solving approach, provides the foundational mindset and skills for formulating and tackling complex problems. This mindset or skills can be acquired using fundamental CT components of decomposing, recognizing patterns in data, sorting and removing complex information through abstractions, and designing algorithms.

Additionally, representations play an important role as they are how information is encoded and processed in AI and ML domains. In AI and ML, representations are crucial for pattern recognition, feature extraction, and knowledge representation. Data science relies on representations to transform raw data into meaningful insights. On the other hand, AI and ML contribute to advancing the state of CT by automating certain cognitive tasks and enabling systems to learn from data. One such example was demonstrated in the carousel repair activity where learners were tasked to program the Sphero robot to visit and repair animals. Learners used functions to automate the robot to visit multiple animals. Through this activity, learners were able to interconnect several representations to visualize concepts. Representations make important support tools in AI by enabling AI systems such as robots understand and process complex data through visualization.

By introducing CT components that relate to emerging topics like AI, ML, and data science into elementary school curricula, educators can equip students with

essential skills and knowledge needed for the future workforce. This study has shown that early exposure to these concepts can enhance problem-solving, and creativity among young learners, preparing them to navigate an increasingly digital and data-driven world. The mutual benefit CT, AI, ML, and data science can afford lies in their power to harness the vast potential of representational data for informed decision-making, problem-solving, and automation, ultimately shaping the future of human-computer interactions and technology applications in education.

Generalizability of programming Sphero robots in early-grade classrooms

While the characteristics specific to Sphero robots provide valuable insights into designing programming environments for young learners, they also offer broader implications for the generalizability of the research beyond Sphero robots. Here are a few ways we can use these characteristics to discuss the generalizability of my findings:

User-Friendly Interfaces: In this study, I emphasize the importance of creating user-friendly interfaces. This characteristic extends beyond Sphero robots to any programmable learning toolkits used in early-grade classrooms. Designers of various robotics or coding platforms can apply principles of simplicity and intuitiveness to enhance student engagement and comprehension of CT concepts universally.

Visual Cues and Abstraction Support: The importance of incorporating visual cues and abstraction support within programming environments transcends Sphero robots. While specific features like programmable LEDs may be unique to Sphero, the broader concept of leveraging visual aids to scaffold CT skills can be applicable to a wide range of educational robotics and coding platforms.

Embodied Practices: This study underscores the importance of embodied practices in programming with Sphero robots. However, the underlying principle of leveraging physical interaction for learning is not confined to Sphero robots alone. Designers can

explore ways to integrate cues for embodied practices into programming environments, irrespective of the specific hardware platform learners are using.

Variety of Representations: The necessity to support diverse forms of representation in programming environments is not exclusive to Sphero robots. Designers of any programmable toolkit can gain from prioritizing multiple modalities of representation, including symbols, visuals, and embodied resources, to cater to the diverse learners and learning preferences of young learners. Balancing the level of scaffolding provided in programming environments is a universal challenge. While this study discusses abstractions and representations in the context of Sphero robots, these principles may apply to any educational technology platform. Designers must find the right balance between providing scaffolded support for comprehension and promoting problem-solving skills across different tools and contexts.

Challenges with Geocentric Nature: Although this study identifies challenges specific to the geocentric nature of Sphero robots, addressing similar challenges related to spatial reasoning and orientation can be relevant across various robotics platforms. Designers can explore ways to support students in understanding directions, angles, and spatial concepts in programming environments regardless of the specific hardware used.

In essence, while the research findings are grounded in the context of programming Sphero robots in early-grade classrooms, the underlying principles and insights have broader implications for designing programming environments across different educational technologies. By focusing on universal design considerations such as user-friendliness, multimodal representation, and scaffolding for learning, educators and designers can apply these findings to enhance CT instruction in diverse educational settings.

Potential Avenues for Future Work

As a follow-up to my research on how representations can support young learners acquire CT skills during robotics/programming tasks, I believe CT and representations hold promise in enhancing how students visualize and understand complex mathematics and science concepts in a curricula context. Potential avenues of future work in CT and representations could include a wide range of research and practical applications. Some potential directions for future work in this domain include:

Development of Pedagogical Approaches: Research on effective pedagogical methods and curriculum design to teach CT and the use of representations across different age groups and educational levels. This might involve designing age-appropriate learning materials and strategies, focusing on practical real-world applications in ways that learners can leverage on the power of representations to create innovative solutions to problems. For instance, framing activities within the Sphero.Math curriculum around an amusement park encouraged learners to link representations with everyday experiences. In the monorail activity, learners were able to recognize patterns within the tasks and relate them to LEDs programmed within the activity, thereby determining "start" and "stop" times for passengers boarding and disembarking from the train.

Interdisciplinary Applications: There is an opportunity to explore how CT and representations can be integrated into various disciplines beyond computer science and mathematics, such as science, engineering, art, and social sciences. In this study, I argue that as students develop CT knowledge in robotics programming

environments, they invent gestures and language until they acquire the right language and conventional gestures to express themselves; I argue that learners transition from concrete to representational to abstract as they develop CT skills, making representational knowledge an important intermediate bridge that connects concrete and abstract knowledge.

Assessment and Evaluation: Future work from this study could involve designing and developing robust assessment tools and methodologies to measure CT skills and the ability to work with representations. This could include assessment techniques using learning analytics methods such as Epistemic Network Analysis (ENA) and other Machine Learning(ML) and AI methods to model and visualize the development of epistemic knowledge networks among learners and other cognitive phenomena afforded by representations and CT.

Child-Robot Interaction: Another potential avenue could be to investigate and generalize how representations are used to mediate interactions between children and robots, particularly in the design of robot interactions, user-friendly interfaces, and user experience (UX) design. When robotics systems collaborate with humans, visualizations support both robots and humans by providing feedback to make better decisions. In robotics activities, humans use visualizations to understand robot interactions with the environment. For example, in the Sphero.Math activities, learners used the “aim” light as a visualization tool on the robot to understand the direction the robot will roll. There is potential to understand other generalizable

design features found on other types of robots that can support interactions for young learners.

Limitations

This study was structured to fit within curricula activities, lacking integration with curricular evaluations, consequently, making it challenging to generalize or apply certain learning claims across districts. Comparative longitudinal studies, comparable to those conducted for standardized curricular subjects, could offer valuable insights to the role of representations in CT skill acquisition. At times, representations varied across iterations of the same activities among the same groups of students, posing potential challenges in establishing and monitoring instances where students used representations to problem-solve. This inconsistency made certain representations unstable in both their interpretation and application, thereby making it difficult to generalize, even within the same activity.

Another limitation of the study stems from the lack of uniformity in teaching methods across all settings. Time constraints posed challenges for teachers and students to fully engage with the activities. The time allocated for Sphero.Math lessons were limited to 45 minutes, which included setup time for activities and devices. One teacher opted for a didactic approach where she explained every detail of the lesson to the students. This teaching style significantly reduced the time available for each activity, leaving learners with insufficient time to complete tasks and sometimes feeling rushed, thereby limiting learners' ability to fully express their problem-solving skills. This highlights the importance of considering time constraints

and instructional approaches in when designing these kinds of implementations in curriculum contexts.

While one classroom setting focused on letting the students, self-express and explore the tools, the other setting had a very traditional teacher-directed instructional style with the teacher discussing the lesson before the students would start the activity. The later style I noticed, impacted how the learners interacted with the tools. In most vignettes, the students mentioned "the teacher said" insinuating their goals were more oriented towards teacher expectations than completing the tasks. Students who had a more experiential play setting without much of a teacher intervention were seen to be more open to exploring creativity and expressing diverse ways to problem-solve the tasks. This highlights the importance of integrating teacher instruction into professional development (PD) sessions, particularly for in-service teachers without experience teaching with computational tools. By providing appropriate PD training and support, teachers can better facilitate student engagement and support learners' creativity in problem-solving tasks, ultimately enhancing the effectiveness of teaching lessons involving computational tools.

Conclusion

The focus of K-12 CT activities has predominantly focused on the development of programming skills, STEM skills or the production and use of other computationally related skills. Much of the research on CT has focused on defining CT and developing and testing curricula. Few studies have focused on aspects of representations and embodiment that can lead to the development and assessment of CT skills using contemporary learning science

methods. In this dissertation, I have demonstrated the role of representations in the development of CT skills in CT and robotics activities in fourth-grade mathematics classrooms. I identified forms of representations, including concrete manipulatives, graphics, symbols, language, and embodied representations, as a lens to understand how students engaged with and navigated activities. My analysis elucidated how each of these representations is either supported or translated to CT skill development. Analysis revealed how representations supported the construction of CT knowledge and worked together to support the development of CT skills and ground disciplinary concepts for learners by complementing and reinforcing each other. These interconnected representational systems tend to be idiosyncratic to each learner or shared by the whole group of learners.

This study supported that when appropriately designed, programmable robots can make effective CT learning tools/environments for young learners. This study's main contribution is to our understanding of how representations can be used as scaffolds to support young learners in learning to "think computationally". This study was able to provide an understanding of how representations embedded in the *concepts* and *designs* of the tools in the learning environment supported young learners' acquisition of CT knowledge. In terms of understanding how CT knowledge is acquired by young learners, this study demonstrates how the use of representations can support young learners to tinker, debug, and refine code in both virtual and physical spaces

based on their perspectives (understanding) of the environment and the practices (engagement) they enact to accomplish tasks.

This study's contextualization of problem-solving in CT environments can be important for teachers, educators, curriculum developers, and educational technology designers to understand ways in which young learners navigate CT environments to make decisions that can support problem-solving and consequently skill development (Wing, 2006). This study was able to explore and establish the relationship between representations and CT, highlighting the multifaceted nature of CT, in robotics environments. One theme that stood out was the interconnected nature of representations that built on each other to make meaningful connections that led to expressions of CT knowledge. These interconnected representations use various forms of data representation, such as graphs, diagrams, symbols, language, embodiment, or visualizations, to demonstrate relationships between elements of a problem or concept.

In CT environments, representations are used to break down complex problems into more manageable components, allowing learners to identify patterns, connections, and relationships between different elements. This approach helps in developing problem-solving skills and enhancing the understanding of computational concepts.

This work contributes to work by scholars like Barr and Stephenson, and participants at the National Research Council (NRC) workshop on the scope and nature of CT

who argued that CT revolves around the process of abstraction, considering how they are formed and their inherent nature.

In this study, I have argued that the Sphero.Math curriculum, coupled with the design of Sphero robots as CT enhancement tools, aligns with the initiatives of the Computer Science Teachers Association (CSTA) and the International Society for Technology in Education (ISTE) to integrate CT into the K-12 curriculum. Given the significant role that abstractions play in CT practice and skill development, this approach supports broader efforts to advance CT education.

Through its design and carefully crafted features, the Sphero robot is an effective educational tool that empowers young learners to abstract and grasp programming concepts, connect them to real-world scenarios, and develop CT skills.

This dissertation advances our understanding of the role representations play in supporting young learners without programming experience to navigate CT environments. Overall, this study demonstrates the potential of representations as a tool for unraveling the dynamics of learner interactions in CT, mathematics, and programming tasks. By visualizing and possibly quantifying the relationships between concepts, representations offer educators and researchers a deeper understanding of the cognitive and social processes that underlie learning in heavily multimodal robotics and integrated CT and mathematics environments. Concepts were linked by either representation that revealed CT knowledge or cognitive processes such as pattern recognition, decomposing tasks, algorithm design, use of coding blocks, code modification, understanding loops, logical thinking, debugging, and abstraction.

In conclusion, the findings of this dissertation strongly support the notion that introducing programming robots in elementary grade classrooms can significantly contribute to the development of CT skills among young learners. CT it has been argued is undeniably a fundamental skill that extends beyond computer science and is essential for navigating our increasingly technology-driven world. CT is recognized as a 21st-century skill, comparable in significance to traditional literacy skills like reading and writing (Wing, 2006). Therefore, as we acknowledge the importance of CT skills, it becomes imperative to explore innovative and inclusive approaches to broaden participation among all learners. To do so, we must ensure that access to equitable learning opportunities in CT education are extended to learners of all ages and backgrounds. By embracing this challenge and designing inclusive practices through activities and curricula like the Sphero.Math curriculum, we can empower future generations to thrive in the digital age and contribute meaningfully to the technology-driven world. In this pursuit, the integration of programming robots in elementary classrooms represents a promising step forward, emphasizing the need to prepare young learners for a future where CT skills are not only beneficial but essential.

Appendix A

<table>
<tr><th colspan="3">Lesson #6: To Ride or Not to Ride</th></tr>
<tr><td colspan="3">Overview
This lesson uses Sphero to check if students are tall enough to go on a roller coaster! This activity involves a pre-built program that will use an <code>if</code> statement to give one of two roller coaster names based on the student's height. Students will need to discuss in groups and as a class to figure out what the code means, how it works, and what the implications are for being able to ride the Raptor roller coaster!

Students were successful in this lesson if they are able to identify whether the program would let someone ride or not, given that person's height.</td></tr>
<tr><td colspan="3">Pre-Built Code: https://edu.sphero.com/remixes/9279379</td></tr>
<tr><td colspan="2">Eureka Module Alignment
Module 2 Lessons 1, 2, and 3</td><td>Prep Time: 5 Minutes
Instructional Time: 45 minutes</td></tr>
<tr><td>Objective</td><td colspan="2">Students will be able to analyze Sphero code to identify its function.
Students will be able to use symbols of inequality to compare measured values to given values.
Students will be able to determine the most likely used measurement unit based on Sphero programming.</td></tr>
<tr><td rowspan="2">Standards and Skills</td><td colspan="2">Common Core State Standards in Mathematics
and Standards for Mathematical Practice
4.MD.1: Know relative sizes of measurement units within one system of units including km, m, cm; kg, g; lb, oz.; l, ml; hr, min, sec. Within a single system of measurement, express measurements in a larger unit in terms of a smaller unit. Record measurement equivalents in a two-column table.
4.MD.2: Use the four operations to solve word problems involving distances, intervals of time, liquid volumes, masses of objects, and money, including problems involving simple fractions or decimals, and problems that require expressing measurements given in a larger unit in terms of a smaller unit. Represent measurement quantities using diagrams such as number line diagrams that feature a measurement scale.
4.NBT.A.2: Read and write multi-digit whole numbers using base-ten numerals, number names, and expanded form. Compare two multi-digit numbers based on meanings of the digits in each place, using >, =, and < symbols to record the results of comparisons..
SMP 5: Use appropriate tools strategically
SMP 6: Attend to precision</td></tr>
<tr><td colspan="2">Computational Thinking Skills
Abstraction
By the end, we have created a program that can be modified to check any height. So this program represents a generalized solution for measure heights (both taller than and shorter than)
Algorithms
We have defined an explicit sequence of instructions for the Sphero to follow in order to accomplish the goal. The algorithm also</td></tr>
</table>

	includes conditional logic, which is a powerful programming tool for creating algorithms

Important Vocabulary

- **Minimum and maximum:** For this lesson students need to understand minimum and maximum in the context of height. They will need to understand what it means to have a minimum height requirement to ride the most intense roller coaster and a maximum height requirement to ride the rides intended for younger kids.

Materials and Preparation

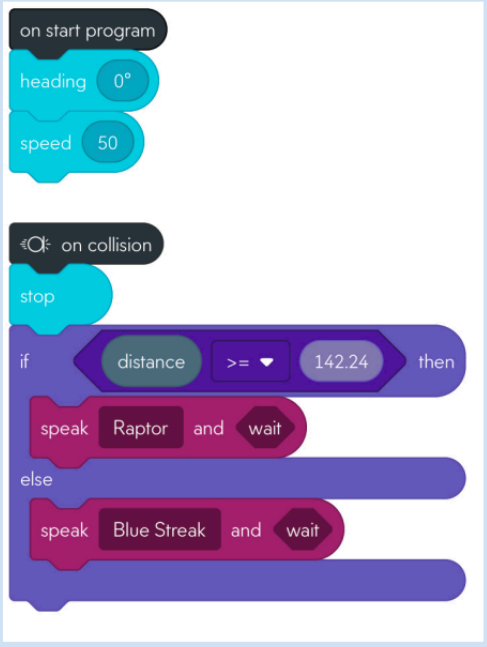
- ❑ 1 Fully charged Sphero and tablet per student group
- ❑ Enough wall space for groups to be spread out evenly
- ❑ String, scissors, scotch or masking tape
- ❑ A meter stick for measuring heights

Lesson Plan

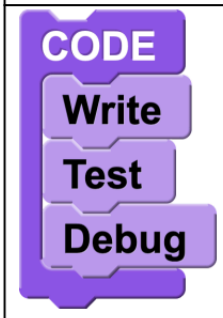
ENGAGE
Decompose
Discuss

Framing:
For safety reasons, some rides have a height requirement. Our amusement park needs a way to measure the height of visitors in ride lines to ensure they can ride.

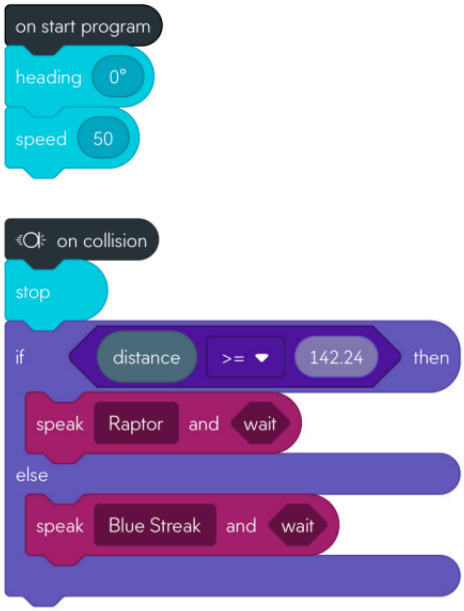
Mission: What height is a safe height? (1 min)
What is the minimum height (with units) you need to be in order to ride all the rides? Sphero can help us with this task using the following program:



Our mission is to figure out what this program does and what height will ensure everyone's safety on the ride.

	Review the Program: (5 mins) In their group, students read the program to determine what they think the program does. ● **Guiding Questions:** ○ What parts of the program do you recognize? What do you think they do in this program? ○ Which parts or blocks of the program are unfamiliar? Based on what the blocks say, can you make predictions about what the commands do? As a whole class, students share their predictions and reasoning for their predictions. ● **Guiding Questions:** ○ You've seen the commands "speed" before. You've also used degree measures in commands before. Compare how this program uses those pieces of code compared to how you've used those pieces of code before. ○ What do you think >= means? ○ What do you think "on collision" means? ○ Why would the code have different commands for different sets of rides? ○ What does distance have to do with determining height in this situation? ○ What are the differences between the rides in the Raptor, and the Blue Streak? **Note: The questions above are meant to get students to make predictions about what unfamiliar parts of the code do and what they should be doing as they try to run the code in the next section. Students should come away from this discussion with some predictions about the code and reasoning behind those predictions. They should also have some idea that distance in the code is the equivalent of height in this context.**
	**Run the Code: (15 mins)** TEACHER MODEL: 1. Cut a piece of string that is equal to your height. 2. Tape one end of the strip to the wall and pull the other end of the string so it is stretched along the floor. 3. Start Sphero at the end of the string away from the wall and run the program towards the wall. In order to figure out what the code does, students can run the code using their own heights. Students should have time to run the code, compare it to their predictions and figure out how it can be used to determine who can go on what rides.

	• **Guiding Questions:** ○ What is the code actually telling the Sphero to do? ○ Do your predictions match Sphero's actions? If so, why do they match? If not, what is different? ○ As students run the code, they should be attentive to the meaning of the IF statement. Questions could include: ■ What does the first line of the if statement mean in everyday language? ■ What do the next 3 lines of code tell Sphero to do? **What Height?: (10 mins)** Once students figure out what the program does, they should also figure out what the cut-off height is to ride the more intense rides. Students might want to use the meter stick to measure their actual height. • **Guiding Questions:** ○ What are the possible units on 142.24? What is that in mixed meters-and-centimeters? ○ Does anyone in their group know their height? How could knowing someone's height be helpful in figuring out the cut-off height? **Extension: (5-10 mins)** Amusements parks can also have rides that are made just for little kids. The ride Kite Eating Tree has a maximum height requirement of 1.37 meters. • Design a program that would help you determine if someone met this maximum height requirement. • A video of "Kite Eating Tree": https://www.youtube.com/watch?v=cHZtLskLnZ8
	**Whole Class Debrief: (10 mins)** Bring students back together to have a class debrief about their experience: • **Guiding Questions:** ○ What's the purpose of an if statement? ○ What if the minimum height was 100 cm? Or 1m and 50cm? How would the code need to change? ○ How did you figure out the units of the height? What units were not reasonable and why? ○ How did you use your code to check whether someone was tall enough or not? ○ To what extent is this a practical use for robots? Would there be a more efficient way to use a robot or another tool? ○ Are there any other pros or cons to using a robot to check customers' height?

Program Key	 on start program heading 0° speed 50 on collision stop if distance >= 142.24 then speak Raptor and wait else speak Blue Streak and wait

Note: Lesson 7 is an extension lesson that is not required to continue with Lesson 8 but highly encouraged.

Lesson #7: To Ride or Not to Ride Extension

Overview

This lesson uses Sphero to collect (and speak!) heights as measured by Lesson 6's strings. It then asks students to plot their measured heights on the board in a summary graph and analyze what measures seem like good metrics for the class data. **Note: This lesson includes discussion on mean, median, and mode which may be outside the scope of your class standards. These discussions are marked, and can be treated as optional depending on what your students are ready for!**

Students were successful in this lesson if they are able to choose a summary metric based on features of a data distribution.

Pre-Built Activity: There is no pre-built code for this lesson (example code included).

Eureka Module Alignment	Prep Time: 5 minutes Instructional Time: 45 minutes

Objective	Students will be able to modify a Sphero program to determine individual height data. Students will be able to use mode, median, and mean to determine best height cut-off using class data.
Standards and Skills	**Common Core State Standards in Mathematics and Standards for Mathematical Practice**
	Computational Thinking Skills • Decomposition ○ We are trying to figure out the cutoff height but in order to do that, we need to know each park visitors' height, so we are breaking down the overall problem (e.g. mean height) into a step that can be used to solve that problem (measuring each person) • Abstraction ○ The resulting program is a generalized solution for measurement that can work regardless of the height of the person. ○ This program is not just for measuring kids but can measure anything; so we have created a generalizable solution to a problem (measuring things) • Algorithms ○ We define a set of steps to measure distance along the ground

Important Vocabulary

Mean: the "average" height, or what the height would be if all the heights were combined together and redistributed so all heights were the same
Median: the height in the middle, if all the heights are put in order from least to greatest
Mode: the height that appears most often (if no heights repeat, there is no mode)

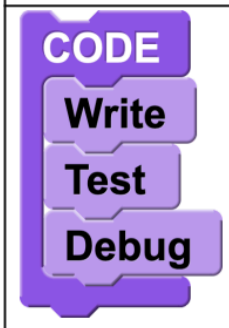
Materials and Preparation

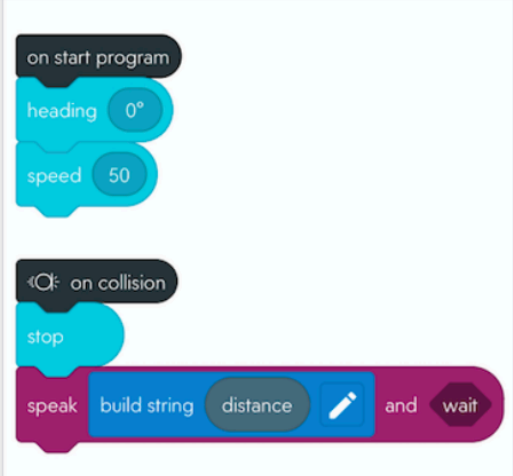
- ❑ 1 Fully charged Sphero and tablet per student group
- ❑ Student string pieces from the last lesson

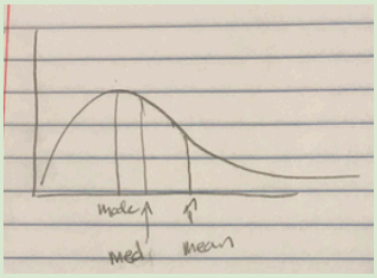
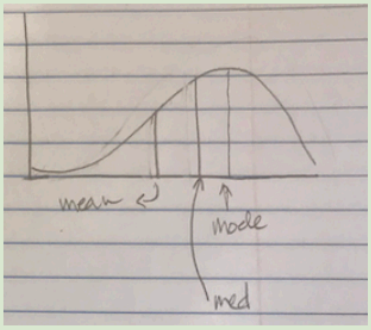
Lesson Plan

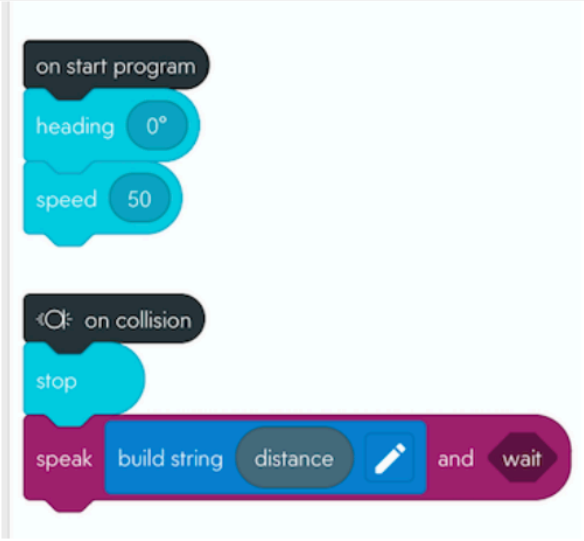
<table>
<tr>
<td> 
</td>
<td>Framing: In the last lesson, we determined that height limited some of the rides that customers can go on. In order for customers to have the best in-park experience, we want to build our rides so as many people as possible (in this case, as many people in our class) can go on them.

Mission: What cut-off height should we design our rides around so that the largest group of people can go on all of the rides?
• Why are there minimum height requirements for rides at amusement parks?

Planning: What information do we need? (5-10 mins)
• Have students spend time brainstorming what they would need to know about customers (the class) in order to determine the proper cutoff height.
○ Possible answers: Everyone's height, how many people there are in the class, the height of the tallest and shortest people, the average height, median height, if any people are the same height.
• Guiding Questions:
○ How can it be helpful if we have everyone's height?
○ Should we use the height of the smallest person? What are the pros and cons of this plan? What about using the height of the tallest person?
○ What are some math measures that might help us serve the most people? What are the pros and cons of these measures? [Note: mean, median, and mode are typically listed in later years with Common Core, so this and next section are optional.]
■ Mean: the "average" height, or what the height would be if all the heights were combined together and redistributed so all heights were the same
■ Median: the height in the middle, if all the heights are put in order from least to greatest
■ Mode: the height that appears most often (if no heights repeat, there is no mode)
○ Can we make a decision about which measure we should use for our height requirement? Why or why not?
■ Possible answer: We first have to figure out everyone's height and what these values are before we can decide.</td>
</tr>
<tr>
<td> 
</td>
<td>Find Class's Heights: (10 mins) (If time is limited, consider measuring heights beforehand)
• Guiding Questions:
○ Last time, we developed a program that told which rides someone could go on based on their heights. Can we use this program to help us determine the heights of each person in our class? Why or why not?
■ This program just tells us if the individual is tall enough to ride certain rides - not their height.
○ What parts of the program can we keep to help us determine someone's height? What do we need to change?</td>
</tr>
</table>

	▪ Students can develop their own program or use the program provided.  1. Distribute the string pieces from last class evenly between groups (it doesn't matter if students get their own string). 2. Use this program (or modify the program from last lesson) to find the length of all the pieces of string. 3. Have students record lengths and share with class.
DEBRIEF Review Reflect Evaluate	**Mean, Median and Mode: (15 mins)** ● **[Note: This question can be skipped as mean, median and mode come up in future years in Common Core, but we are including it in case you want to have the discussion.]** Once students have all of the heights, they should determine the class mean, median and mode (use calculator or have the teacher calculate mean). ○ Mean: the "average" height, or what the height would be if all the heights were combined together and redistributed so all heights were the same ○ Median: the height in the middle, if all the heights are put in order from least to greatest ○ Mode: the height that appears most often (if no heights repeat, there is no mode) ○ **Note: If students are not familiar with these measures, you might have to give them more explicit math instruction (i.e. give them the formula to find the average).** ● Create a graph of student height data (could use Excel or another graphing program to complete quickly, but it is recommended to give students a way to plot their own heights in a dotplot or histogram). Show where mean, median, and mode would appear on the graph. ● Students can then determine which cutoff heights (the mean, median or mode) would be most inclusive of all of their all of their classmates.

	Whole Class Discussion: ● **Guiding Questions:** ○ Which measure should we design our rides around? ○ Why wouldn't we want to automatically choose the lowest? The highest? ○ Depending on the make-up of your class, this answer could change: ■ If distribution is skewed right (i.e. there are some really tall outliers), median or mode will likely be the best because outliers are pulling the average to the right.  ■ If distribution is skewed left, then mean is going to be best because outliers pull average down, allowing more "heights" to ride rides.  ■ If distribution is close to normal (which is possible) then measures should be pretty close and any measure would work. ○ Does our data make sense to make decisions on ride design for the whole amusement park? ○ What are some of the pros and cons? ○ Think about the program we created to measure height. What does this program do? ○ What other uses can you think of for this program?
Program Key	

	 Make sure to change "**build string**" box to a number format to insert distance block.

Lesson #8: Monorail Stops

Note: This lesson is a 2-day lesson, with Day 2 being optional but strongly encouraged because it shows the value of using loops to code more efficiently.

Overview

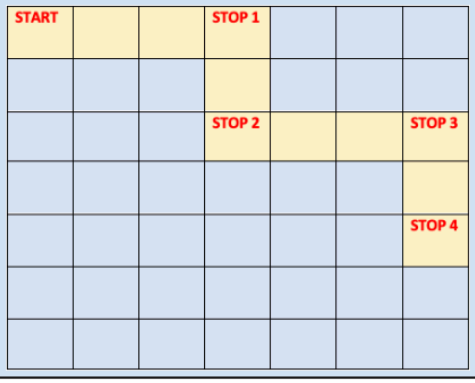
This lesson challenges students to build a program to guide Sphero through a fixed path by changing angle, speed, and time in the `roll` block. On the second day, students expand and re-work this program to be more efficient using a `loop` block!

Students were successful in this lesson if they are able to identify whether the program would let someone ride or not, given that person's height.

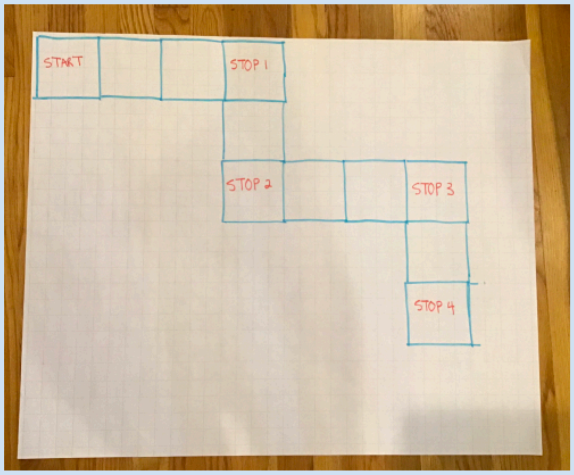
Pre-Built Code: This lesson does not include a pre-built code activity.

Eureka Module Alignment	Prep Time: 30 minutes
Module 3, Lesson 25; Module 4, Lessons 5-8	Instructional Time: Day 1: 45 minutes Day 2:1 hour

Objective	Students will code Sphero to navigate a fixed path by manipulating the angle, speed and time in the roll block Students will understand the relationship between turning in different directions and angles. Students will find opportunities to code more efficiently by looking for patterns in the context of problem. Students will create loops using patterns from the provided map and the heading block.
Standards and Skills	**Common Core State Standards in Mathematics and Standards for Mathematical Practice** **4.MD.5:** Recognize angles as geometric shapes that are formed wherever two rays share a common endpoint, and understand concepts of angle measurement: **4.MD.C.5.A** An angle is measured with reference to a circle with its center at the common endpoint of the rays, by considering the fraction of the circular arc between the points where the two rays intersect the circle. An angle that turns through 1/360 of a circle is called a "one-degree angle," and can be used to measure angles. **4.MD.5B:** An angle that turns through n one-degree angles is said to have an angle measure of n degrees. **SMP 5:** Use appropriate tools strategically. **SMP 6:** Attend to precision.
	Computational Thinking Skills • Decomposition ○ Breaking down the segments of the monorail path to be followed.

	• Algorithm ○ Defining each of the steps to be followed to move the monorail and representing them programmatically • Pattern Recognition ○ Identifying repeating structures/distances/commands that can be reused. Examples include noticing that the monorail path goes a certain number of blocks repeatedly or that the code for one grid block can be used over again. • Programming ○ Writing the program to be carried out by the Sphero.
Important Vocabulary **Efficient:** students will have several discussions about the most efficient way to program the robot. Looping will be highlighted as a way to program a repetitive task more efficiently. i	
Materials and Preparation ❑ Grid maps for each group of students - Using chart paper with a grid, the teacher can create maps for each student group beforehand. The teacher could also have students draw the maps during the lesson. ❑ 1 Fully charged Sphero and tablet per student group	
Lesson Plan	
ENGAGE Decompose Discuss	**Day 1** **Framing: (Whole Class Discussion) (1 min)** Many theme parks are big enough that it would require a lot of walking to get from one attraction to another. One way to move park visitors through the park efficiently is with a monorail. Today, we're going to use Sphero as our monorail and program it to drive the monorail path through our park. **Mission: (2 mins)** • Give the **grid mat** to students. • Sphero needs to move from the Start block Stop 4, stopping at each "bus stop" along the way and flashing its lights from blue to green. **Map of Grid:** 

Note: Grid map might look something like this:



Planning: (5 mins)
Once students have their grid, give them time to plan how they will program their monorail path.

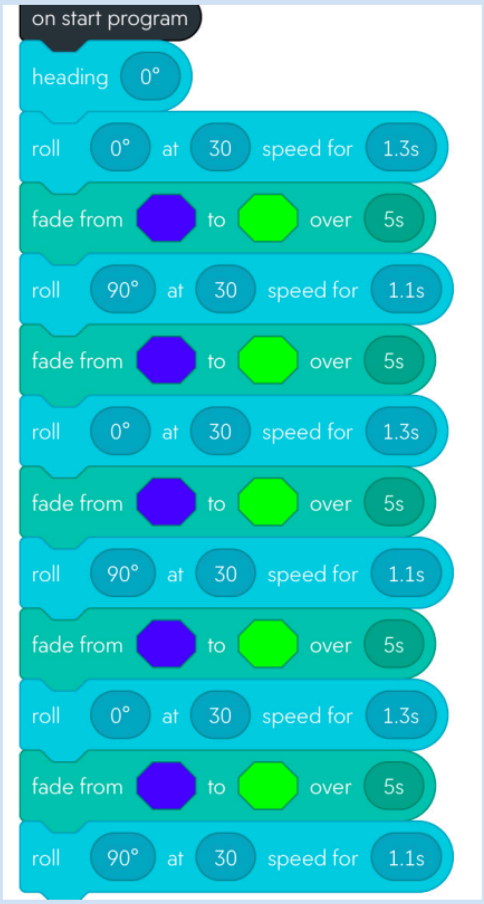
- **Guiding Questions (while planning):**
  - What blocks will we need to use? (Roll block, light block)
  - What order will the blocks need to be in?
  - What will we need to know to program the roll block? (how far between each stop, how many degrees Sphero needs to turn)
  - Do you see any patterns in the path? If so, how can these patterns be helpful for programming?

Whole Class Discussion: (5 mins)
Once students have created a problem-solving strategy and decided which code blocks they will need to use, they will present their solutions to the class.

- After each group presents their strategy, the class should provide feedback using **"I wonder if"** statements.
- Ask students to compare programs, does any program seem more efficient than the others, what parts of the code seem really useful?

	**Coding Creation/Testing/Debugging: (15 mins)** ● Provide time for students to determine how to program the roll block to move between each "stop". ● Circulate among students and help them as they test and debug their programs. ● **Guiding Questions:** ○ What do you need to change if your Sphero is going a little too far or a little too short? ○ What do you need to think about when making Sphero turn? ○ How does that affect how you code the Roll block? ○ How can you make Sphero move one grid square? ○ How is that helpful for coding the rest of the program? ○ Do you see any patterns in the path? How could that be helpful in your coding ■ Possible Answer: Students have to move 4 grid squares and 3 grid squares two times each. They should be able to re-use their code.
	**Whole Class Debrief: (10 mins)** ● **Guiding Questions:** ○ How did you determine the different distances that Sphero had to travel? ○ What variables can you change to make Sphero go a little farther or a little shorter? Do you think one variable or another is better to change? Why? ■ Answer: Students can change either time or speed to affect Sphero's distance. ○ What did you learn about changing the direction with Sphero? ○ What is the relationship between the angle measures when Sphero turns corners? ○ Were you able to re-use any code? How did you know when you could reuse code? ○ Did you have to debug Sphero? If so, how?

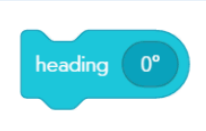
- One student created the following code to navigate the new monorail path:

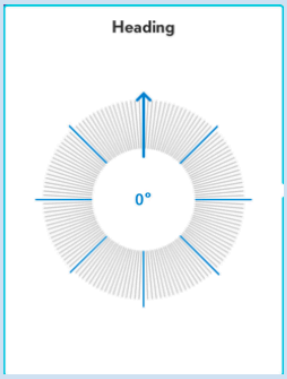


- **Guiding Questions:**
  - What are the advantages and disadvantages of this code?
  - Are there any patterns you see in this code?
  - Is there a way to make this code more efficient?
    - Answer: Yes, we can use looping.

Today, we'll figure out how to make our Sphero monorail path more efficient by using looping.

Introduce the Heading block. (5 mins)
In order to use a loop in this situation and be able to make Sphero turn, students will have to have some familiarity with the heading block and how angles work for turning Sphero





- **Guiding Questions:**
  - Where have you seen heading before?
  - What do you think heading does?
  - What angle measures make Sphero go left, right, forward and backward?
  - What's the relationship between these angles?

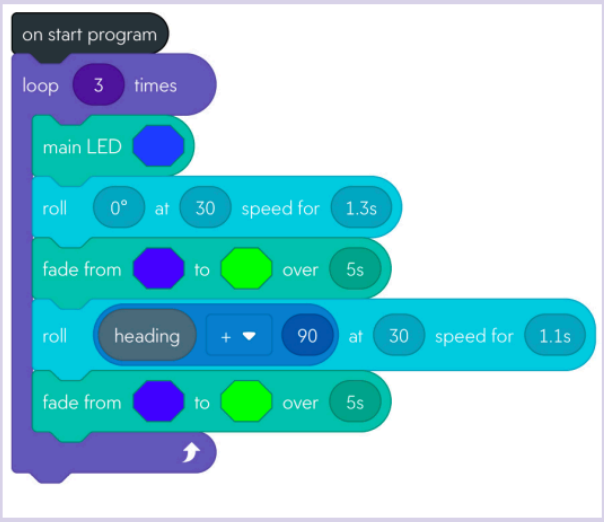
Roll block for loops (5 mins):
Students will be given the code for the roll block that can be put into the loops. Spend some time making predictions about what this code does.

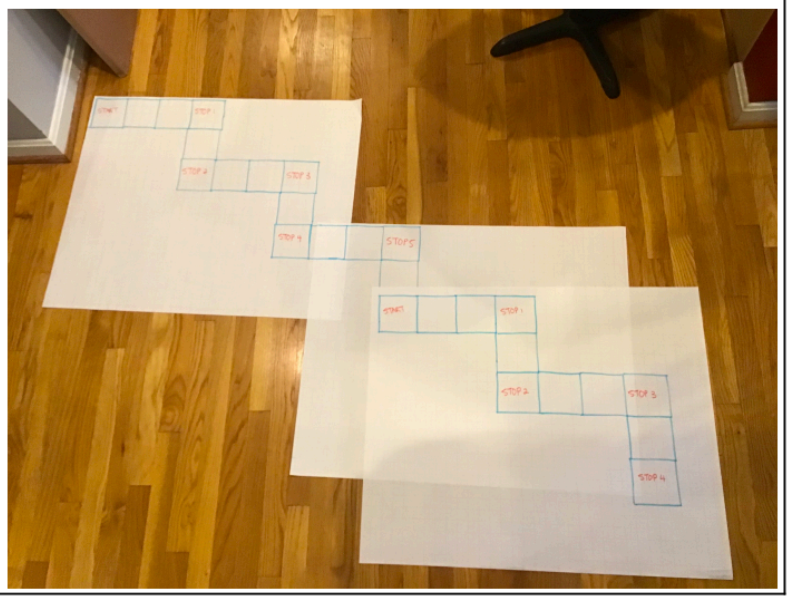


- **Guiding Questions:**
  - What do you think this command does?
  - What parts of these commands are familiar? How can they help us figure out what these commands do?
  - Note: It's not essential that students know exactly what these commands do right now, but discussing them first can give them some ideas when they use them in their loops.

Planning: (10 mins)
Give students time to plan out their new program with the new grid map.

- **Guiding Questions:**

	○ How is this code different from your day 1 code? Are there any parts of the code that you can re-use? ○ How many grid squares do you have to move each time? How does that help you with looping? ○ How many times should your loops loop through each command? ○ What angle directions are you going to turn? How do you know? **Whole Class Discussion: (5 mins)** Once students have created a problem-solving strategy and decided which code blocks they will need to use, they will present their solutions to the class. ● **Guiding Questions:** ○ After each group presents their strategy, the class should provide feedback using "I wonder if" statements. ○ Ask students to compare programs, does any program seem more efficient than the others, what parts of the code seem really useful?
	**Coding/Testing: (20 mins)** ● Give students starter code and provide time for students to determine how to program two different loops to make Sphero travel through the monorail route. ● Circulate among students and help them as they test and debug their programs. ● Final Program should look similar to this:  ● **Guiding Questions:** ○ How is this code different from your day 1 code? Are there any parts of the code that you can re-use?

	○ How many grid squares do you have to move each time? How does that help you with looping? ○ How many times should your loops loop through each command? ○ What angle directions are you going to turn? How do you know?
	**Whole Class Debrief: (10 mins)** • **Guiding Questions:** ○ What patterns were helpful in creating loops? ○ What was important for angles and looping? ○ How is the code you created more efficient than the code from the student at the beginning? ○ Why were we able to use loops for this monorail route? What was different from day 1? ○ Were you able to reuse any code from day 1? ■ How did you know when you could reuse code? ○ Which code applies to the even bus stops and which code applies to the odd bus stops? How do you know?
Extension	To show the power of looping, multiple student groups could connect their grid maps and then make Sphero travel through all of the maps. This can be accomplished by lining one groups "START" with another groups "STOP 6" as shown here: 

Program Key	 roll heading + 90 at 30 speed for 1.1s on start program loop 3 times main LED roll 0° at 30 speed for 1.3s fade from to over 5s roll heading + 90 at 30 speed for 1.1s fade from to over 5s

Lesson #9: Carousel Repair

Note: This lesson is a 2-day lesson.

Overview

This two-day lesson challenges students to modify a program to guide Sphero to equally-spaced animals on a "carousel" using angles on a circle. In the second day, they modify a more efficient version of this code which uses a function to compress the visit-and-return path into a single line of code.

Students were successful in this lesson if they are able to use knowledge of angles to move Sphero to correct locations on a circle, and use a function to repeat this code efficiently for multiple iterations.

Pre-Built Code:

Day 2: DCPS Carousel Repair

Eureka Module Alignment Module 4, Lessons 5, 8, 10, 11	Prep Time: 20 minutes Instructional Time: 50 minutes

Objective	Students will be able to command Sphero to move through different angle measures between 0 and 360 degrees in 45 degree increments. Students will notice the collection of angles that Sphero moves through to create points on a circle. Students will recognize that Sphero moves at angles relative to its original orientation They will be able to account for that orientation to command Sphero to move backwards through they angles that they originally created
Standards and Skills	**Common Core State Standards in Mathematics and Standards for Mathematical Practice** **4.MD.C.5.A:** An angle is measured with reference to a circle with its center at the common endpoint of the rays, by considering the fraction of the circular arc between the points where the two rays intersect the circle. **4.MD.C.7:** Recognize angle measure as additive. When an angle is decomposed into non-overlapping parts, the angle measure of the whole is the sum of the angle measures of the parts. **4.MD.5B:** An angle that turns through n one-degree angles is said to have an angle measure of n degrees. **SMP 4:** Model with mathematics.
	Computational Thinking Skills ● Pattern Recognition ○ After figuring out how to get to animal 1, 2, and 3 - then look for pattern across those three solutions that could tell them how to get the rest (animals 4-8) ● Problem Decomposition ○ Breaking down the steps of visiting each animal ● Abstract ○ Creating a generalizable program that can visit any animal after being told

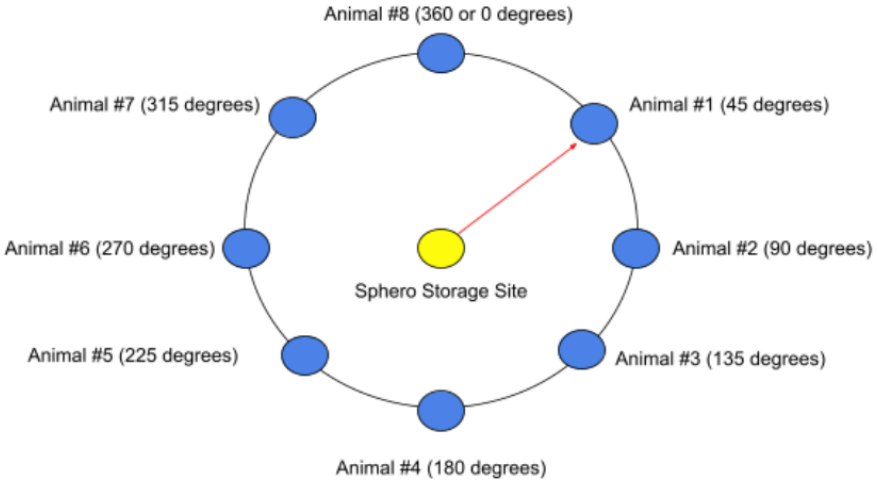
	▪ Create a function called “visit animal _” that just calculates angles • Algorithms

Important Vocabulary

Function: At the end of this lesson students will see their first program written as a function. Functions can be helpful because they allow for different inputs (or variables), which can change the actions of the robot in predictable ways. In this lesson, the function at the end makes the code more efficient and allows users to get Sphero to travel to any of the 8 carousel animals.

Materials and Preparation

- ❑ 1 Fully charged Sphero and tablet for each student group
- ❑ One set of angle markers for each group already marked out on the floor (see diagram). The yellow dot stands for the center of the carousel and the blue dots stand for the carousel animals.
- ❑ Protractors for each group



Lesson Plan

ENGAGE
Decompose
Discuss

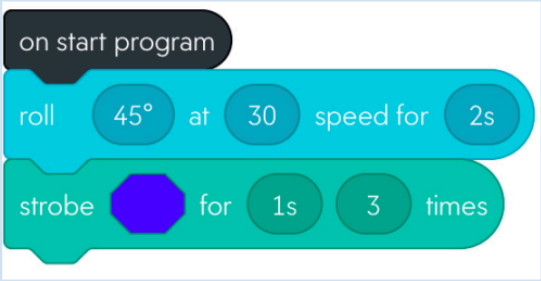
Mission: (15 mins)

There are times when rides at the amusement park break down. A robot could be helpful in quickly getting to the broken parts of rides and fixing them. For example, a robot could be helpful in quickly getting to and fixing broken animals on a carousel so that more visitors can enjoy the ride.

Today’s mission is to figure out how to program Sphero to move to each animal on the carousel to fix it if needed. Each group has a diagram of the carousel that shows where the carousel animals are located. Animals are EQUALLY SPACED around the carousel. Your job is to code Sphero to move from its storage site to carousel animals 1, 2, and 5. Once Sphero reaches the carousel animal, it should be able to retrace its steps and make it back to its storage site.

Give students a chance to look at the diagram. Explain to students that starting from the yellow dot (the storage site), their goal is to create a code sequence that will first

send sphero to each animal, flash blue, and then retrace its path back to the yellow dot. Give students the initial code to help them with the first angle:



Give students time to run the code. Then ask students to try to code Sphero to roll back to its storage site.
Note: By the end of the lesson, students should figure out that adding 180 degrees to any degree measure will get Sphero to move backwards. So if Sphero moves toward animal #1 at 45 degrees, it would move back to the storage site at 225 degrees.

Once students think they have figured out how to make Sphero move backwards from animal 1, have them try to make the full code for animals 2 and 5. As groups develop their plans, the teacher should circulate between groups and prompt them to explain why they think their plan will work. Plans should include measuring the angles Sphero will need to turn to execute the program, understanding what parts of the code they can keep and what parts they will have to change, and thinking about how angles will change when Sphero has to move backward.

- **Guiding Questions:**
  - What does this code tell Sphero to do?
  - What do you notice about when you roll forward from the yellow dot to the blue dot versus the blue dot to the yellow dot?
  - How do you think your code will compare when you move backward versus forward? What will have to change and what can stay the same?
  - What do you think you are going to have to do to move to the next animal? What parts of the code that you add will have to change from the original and what parts can be copied from the original?
  - What do you think the angle for the next animal will be? Why?

Discussion: (5 mins)
Once students have created a problem-solving strategy and decided which code blocks they will need to use, they will present their solutions to the class.

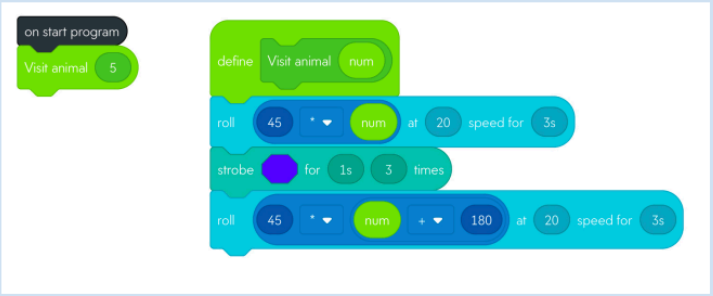
- After each group presents their strategy, the class should provide feedback using “I wonder if” statements.
- Ask students to compare programs, does any program seem more efficient than the others, what parts of the code seem really useful? How are groups

	thinking differently about coding the different angles that Sphero will have to move through, both forwards and backwards? • Is the code that students are thinking about efficient?
	**Code/Test/Debug (15 mins)** Students should create their codes on Sphero programming canvas, and practice running different codes to move Sphero to the different dots. • **Guiding Questions to ask as students work:** ○ What do you notice as you try to get Sphero to move from the center of the carousel to the different angles? ■ Is there a connection between the angles that Sphero has to travel to get to each animal? ■ Can you predict what the next two angles will be? ■ What do you think the final angle will be and why? ○ How is moving backwards different? How can you use the heading menu in the roll command to help you move backwards? ■ Do you notice a relationship between the forward angles and backward angles in each segment of Sphero's route? ○ Code length: ■ Students who are trying to use one program to code all of the angles might find that their code is getting really long. ■ What changes can you make to the code you already have so you don't have to keep adding blocks? ■ Do you have to have all of the code on the "on start program" at the same time? Is there a way to break up the code?
	**Bring students back together to have a class debrief about their experience: (15 mins)** • **Guiding Questions about the overall coding experience:** ○ What did you need to know to code Sphero to move between the markers? ○ How did the commands change as you moved from animal to animal? What commands remained the same? • **Guiding Questions about angle relationships:** ○ How did the relationships between the angles help you with coding? ○ What might change in the code if the carousel were bigger? Why? ○ What would have to change in the code if there were more animals on the carousel? • **Guiding Questions about code evaluation:** ○ Is this code efficient? Why or why not? ■ Some students might have re-typed the code for each angle, other groups might have used one program, making the code long and unwieldy. ○ What part of the code did we use over and over? Might there be a way to make the code shorter?

DAY 2: Functions

ENGAGE
Decompose
Discuss

The following code, which uses a function, behaves similarly to our code from yesterday:
(Have students open the activity DCPS Carousel Repair to access the code on their Ipad)



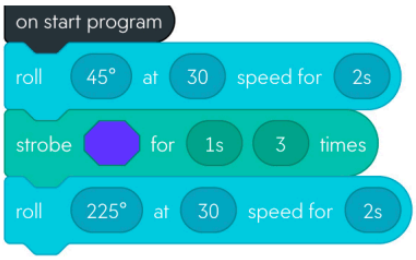
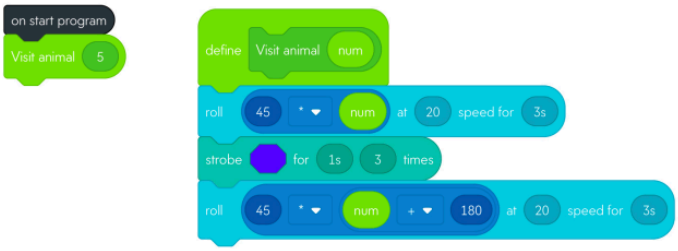
- Explain to students that this code is known as a function and does the same thing as the codes that they made, in this case Sphero goes to animal 5, blinks and returns to its storage site.

CODE
Write
Test
Debug

- Give students a few minutes to look at this code.
- Explore the following:
  - Try changing `visit animal (5)` to another number.
  - What changes? What stays the same?
  - What happens if you add another `visit animal` below the first?
  - How does this code work? What steps does it follow?
  - Where can you see the steps of the code?
  - Can you make Sphero visit animals 1, 2, and 5, one after the other?

DEBRIEF
Review
Reflect
Evaluate

- **Guiding Questions:**
  - What should you change in this code to make Sphero go to different animals? What can remain the same? Why?
  - What does num stand for? How is it working in the code?
    - Note: You can tell students that this part of the code is known as a variable. A variable is helpful because we can assign different numbers to it to make the program run slightly differently (Students also saw variables in Lesson 9).
  - Why is it important to have the "+ 180?" What does this part of the code do?

	■ Note: Students may be confused by the "+ 180" it will cause some of the angles to be greater than 360 degrees. For example, for the code above 5*45+180=405. This might be an opportunity to have a conversation with students that angles can be bigger than 360 degrees, it just means they have gone more than one circle rotation (also known as coterminal angles). ○ Note: Another important math concept to highlight is that multiples of 45 make the angles. Good questions to ask might be: ■ What does "45 * num" mean? Why does that get us to the angle we want? Could we use another operation (addition) to do the same thing? Which operation do you think is better and why?
Program Key	 

Lesson #11: Triangles

Overview

This lesson challenges students to revisit code from the Park Beautification lesson and use that as a jumping off point to write a program that draws an equilateral triangle. Because of how Sphero's internal compass interprets angle headings, however, this lesson also requires converting from the desired interior angle measurement to the *exterior* angle measurement that Sphero needs as input. As with the previous lesson, students may use loops or repeated code to successfully draw the triangle.

Students were successful in this lesson if they are able to build off previous code and a knowledge of how Sphero interprets headings to write a program that makes Sphero draw an equilateral triangle.

Pre-Built Code:

No pre-built code for this lesson.

Eureka Module Alignment	Prep Time: 20 Minutes
Module 4, Lessons 13-14	Instructional Time: 30 - 45 Minutes

Objective	Students will determine how to draw out an equilateral triangle using the Sphero roll command and sensory data.
Standards and Skills	**Common Core State Standards in Mathematics and Standards for Mathematical Practice** **4.G.A.2:** Classify two-dimensional figures based on the presence or absence of parallel or perpendicular lines, or the presence or absence of angles of a specified size. Recognize right triangles as a category and identify right triangles. **4.MD.C.5:** An angle that turns through n one-degree angles is said to have an angle measure of n degrees. **4.MD.C.6:** Measure angles in whole-number degrees using a protractor. Sketch angles of specified measure **SMP 1:** Make sense of problems and persevere in solving them. **SMP 4:** Use appropriate tools strategically
	Computational Thinking Skills ● Decomposition ○ Breaking down the drawing of the triangle into steps ● Pattern recognition ○ Repeating sides of the triangle ○ Similarity between square program and triangle program ● Algorithms ○ Defining steps to create the triangle

Important Vocabulary

Equilateral triangle: For this lesson, students will need to know the properties of equilateral triangles (i.e. equal sides and equal angles).

Materials and Preparation

- ❑ Masking tape or centimeter tape, rulers, protractors
- ❑ Outlines of equilateral triangles taped to the floor. Alternatively, if time permits, you can allow students to tape triangles onto the floor themselves.
- ❑ 1 Fully charged Sphero and tablet per student group

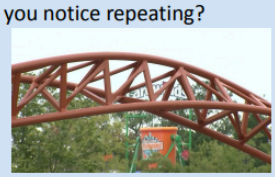
Lesson Plan

ENGAGE

Decompose

Discuss

Framing: Take a look at these rides from various amusement parks. What shape do you notice repeating?







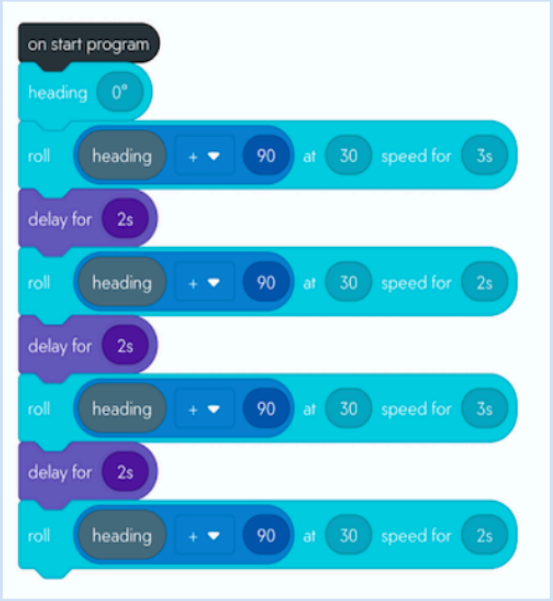


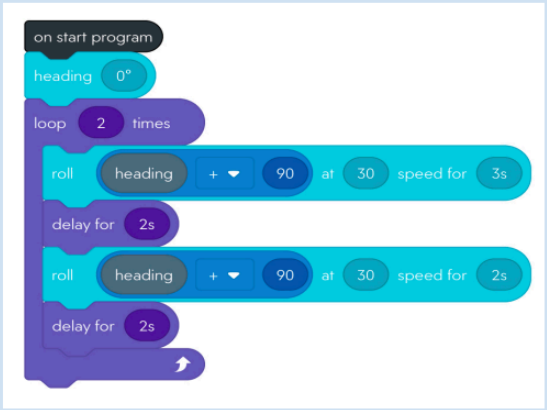
Lots of structures use triangles because they give the structure strength and stability.

Mission: Our mission today is to design a program so Sphero can travel along the triangles of the different rides in an amusement park in order to clean the structures.

- **Guiding Questions:**
  - Why would the rides need to be cleaned?
  - What might be on the structure that we would need to clean it?
  - Why would we want a robot to clean the rides instead of a person?
  -

Think back to the Park Beautification in Lesson 11. In that challenge we used a program to draw rectangles and squares. Let's reflect on two programs from that lesson to think about drawing a triangle:

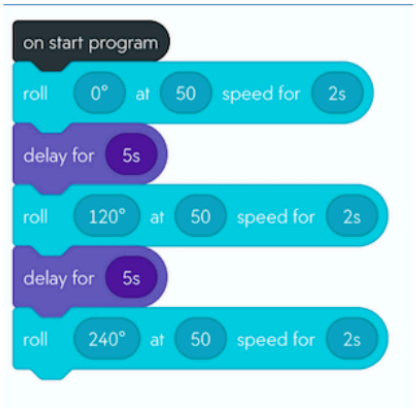


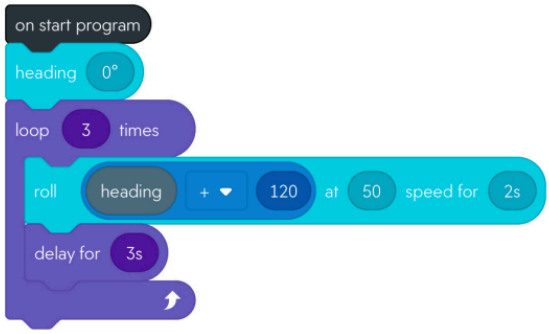


With a partner, discuss the logic of the code.

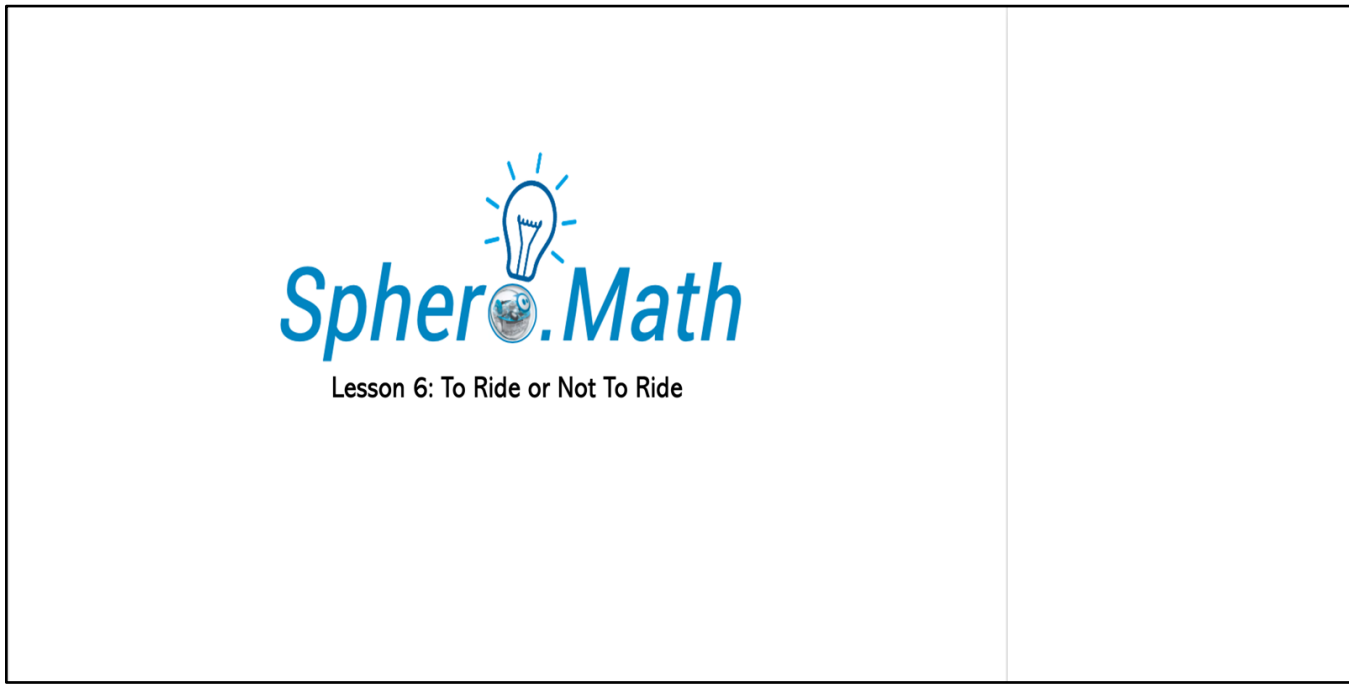
- **Guiding Questions:**
  - What do each of these programs do?
  - Why do these programs draw rectangles?
  - What's the difference between these two programs?

	Now think about drawing a triangle. • **Guiding Questions:** ○ What is the difference between a triangle and a rectangle? ○ How is that going to change the code? ○ How can we build on our program of a square to draw an equilateral triangle? ○ How are squares and equilateral triangles similar? How are they different? ○ If you are making Equilateral triangles what SHOULD each roll degree, speed and distance be? ○ What blocks will you use? ■ While the sides of the equilateral triangle should be coded in a similar way as a square, the angles will be different. Because an equilateral triangle does not have 90 degree angles, students will need to figure out how to code an interior angle of a triangle. This will involve thinking about how many degrees Sphero will need to deviate from its straight path to make an interior angle of 60 degrees. ■ Students will also get some sense that the exterior angles of a polygon have to add to 360 (i.e. Sphero has to travel through a full 360 to get the starting point of a shape.)
	**Teacher tapes various equilateral triangles on the floor ahead of time for groups** Students should spend some time trying to code Sphero to trace the taped triangle. Have students first make a brief plan about what they think the code should be. Once students make their code they should test it out. Give students 10 minutes to test out their code and try to debug. (The sensory data can be a good tool for debugging, students can use it to see how the path the Sphero is actually drawing and compare it to what they want to happen.) ` After 10 minutes, have a discussion with students about what is working and what is not. Have students compare their current programs. • **Guiding Questions:** o What angle were you expecting to use to trace the triangle? ▪ Why is this angle not working? o What angle does Sphero make when it moves straight ahead? o What do we know about coding the sides of an equilateral triangle? o Where did Sphero go compared to where you expected it to go? Paper Options to use for sketch before Debug - sketch the triangle you are trying to make Send students back to coding and debugging for another 10-15 minutes. Eventually, some students should figure out that instead of 60 degrees, Sphero will have to

	travel 120 degrees to make a 60 degree interior angle. The same will happen with the second angle except it will have to be 240 degrees to get back to the starting point. Alternatively, students could use the heading command and add 120 each time. Extensions: ● Have students try to code different types of triangles. (Isosceles, right, make their own). ● Have students figure out how to make an equilateral triangle with a loop ● Can you make a pinwheel design by having Sphero change its heading and using the loop function?
DEBRIEF Review Reflect Evaluate	Have students discuss in groups, as a class, or record in a journal some of the following questions. ● **Guiding Questions:** ○ Why do you think that the angle measurement of the actual triangle is different from the angle measurement that you coded? ○ Why is angle measurement not a problem with the square? ○ Did you notice any patterns in the angle measurement of the actual triangle and the angle measurements you coded? ○ Discussion between theory and practice – why didn't 60 degrees work the first time? ■ Constraints of coding language Exit Ticket: ● What angle might you need to use to make Sphero turn for a 40 degree angle? 45 degrees? 30 degrees? 110 degrees?
Program Key	**Program will vary depending on the triangle. Sample Program key below:** Code without loop:  Code with loop:

	 on start program heading 0° loop 3 times roll heading + 120 at 50 speed for 2s delay for 3s

Sphero.Math Slides



ENGAGE
Decompose
Discuss

Framing

For safety reasons, some rides have a height requirement. Our amusement park needs a way to measure the height of visitors in ride lines to ensure that they can ride.

- Why do you think amusement parks set height requirements for different rides?

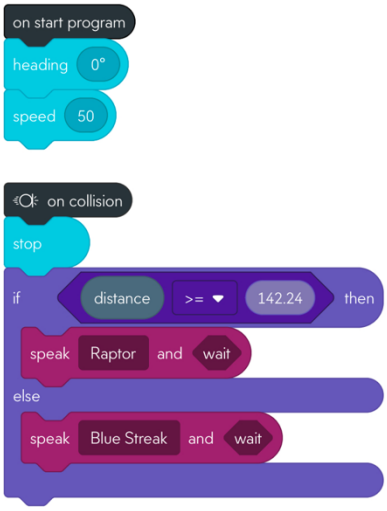


ENGAGE
Decompose
Discuss

Mission: What height is safe?

What is the minimum height that a visitor needs to be in order to ride all rides at our park? Sphero can help us with this task using the program at the right.

Our mission: To figure out what this program does AND to figure out the minimum height requirement to be able to ride all rides.

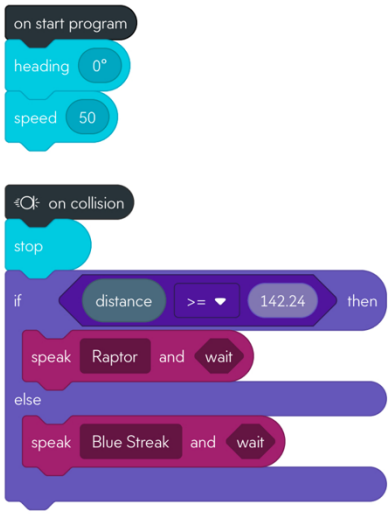


ENGAGE
Decompose
Discuss

Decompose

Discuss what you think this program does with your group.

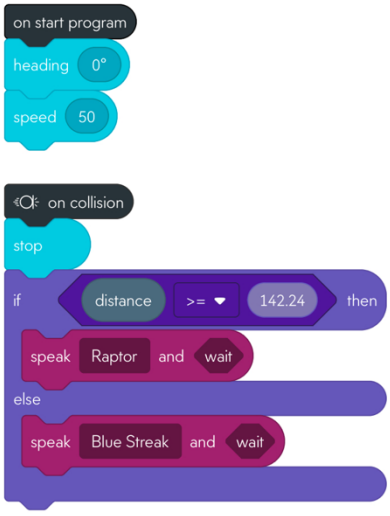
- What parts of this program do you recognize? What do you think they do in this program?
- Which parts of the program are unfamiliar. Can you make predictions about what these blocks do?



ENGAGE
Decompose
Discuss

Discuss

What predictions did your group make about the code?





Test

Use this time to test your predictions and run the code against your group members' heights.

- Do your predictions match Sphero's actions? If so, why do they match? If not, what is different?
- What does the IF statement do?



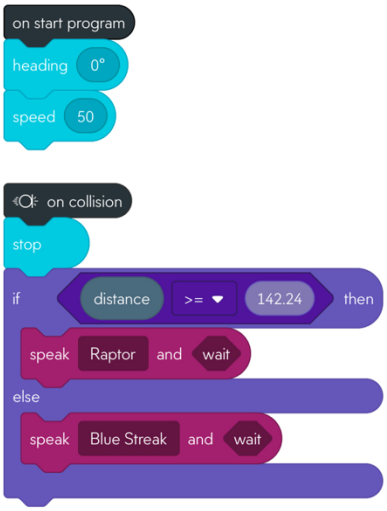
Test: What height?

Now that you know what the code does, what is the minimum height to ride all rides at our park?



Review

- What does this program do?
- What is the purpose of an IF statement?
- How did you figure out the units for the height? What units were reasonable/not reasonable? Why?





Evaluate

- To what extent is this a practical use for robots? Would there be a more efficient way to use a robot or another tool?
- Are there any pros or cons to using a robot to check visitors' heights?

Sphero.Math

Lesson 7

ENGAGE
Decompose
Discuss

Framing

In the last lesson, we determined that height limited some of the rides that visitors could go on. We want our park to be as inclusive as possible and give people the best in-the-park experience. As a result, we want to design rides that allow as many people as possible (in this case, as many people in our class) to ride.



ENGAGE
Decompose
Discuss

Mission

What cutoff height should we design our rides around so that the largest group of people can go on all the rides?



ENGAGE
Decompose
Discuss

Decompose

In your groups discuss:

- What information about customers would be helpful to determine our cutoff height?



Discuss

- How can it be helpful to have everyone's height?
  - Should we use the smallest person's height? The tallest person's height? Why or why not?
- What are some math measures that might help us with this task?
- Can we make a decision about which of these measures to use?



Test

Use our new program to find the heights of everyone in your group and record them for the class.

CODE
Write
Test
Debug

Write

Last lesson, we used this program to determine who could go on particular rides.

- Is this program helpful for our task today? Why or why not?
- What changes do we need to make to this program to complete our task today?



DEBRIEF
Review
Reflect
Evaluate

Find mean, median, mode

Use the whole class data to calculate mean, median and mode.



Reflect

- Based on our data, which measure (mean, median or mode) is best for us to use for our cut-off height?
- Why would we not automatically choose the highest or lowest height?



Evaluate

- Does our data make sense to make decisions on ride design for a whole amusement park?
  - What are some of the pros and cons of using the height we found?
- Think about the program we used to measure height? What does this program do?
  - Can you think of other potential uses for this program?



Lesson 8: Monorail Stops

ENGAGE

Decompose

Discuss

Mission

A monorail would be helpful to visitors in our park. Here is a map of our monorail path, and we need to program the monorail to run the path. We can use Sphero as a model to help us program.

Sphero should do the following:

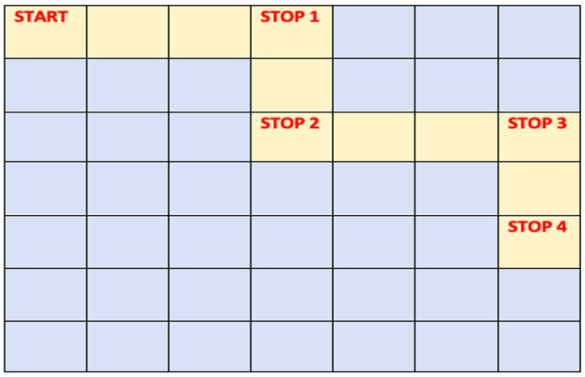
- Navigate to each stop.
- Wait for three seconds for passengers to get on.
- Blink from yellow to green to indicate that passengers can get on.

START			STOP 1			
			STOP 2			STOP 3
						STOP 4

ENGAGE
Decompose
Discuss

Decompose

- What blocks will we need?
- What will we need to know to program the roll block?
- Do you see any patterns in the path? If so, how can these patterns be helpful for programming?





Write/Test/Debug

Take some time to write and test your program. Debug when necessary!



Discuss

Compare other groups' coding ideas with your own.

- Do any parts of groups' codes seem more efficient?
- What parts of the code seem really useful?



Distance

- How did you determine the different distances that Sphero had to travel?
  - What variables can you change to make Sphero go a little farther or a little shorter? Do you think one variable or the other better to use? Why?



Angles

- What did you learn about changing direction with Sphero?
  - What is the relationship between angle measures when Sphero turns corners?



Efficiency

While you were coding, were there any patterns in the task that made coding easier?

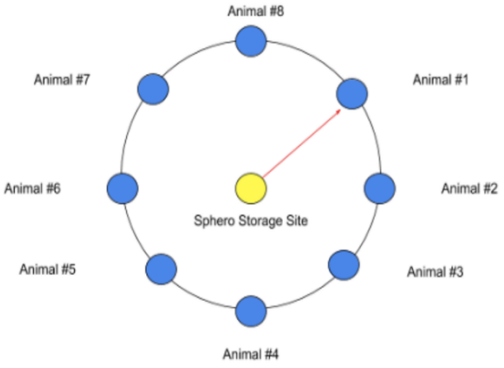
Sphero.Math

Lesson 9: Carousel Repair

ENGAGE
Decompose
Discuss

Mission

- Today's mission is to figure out how to program Sphero to move to each animal on the carousel to fix it if needed. Your job is to code Sphero to move from its storage site to carousel animals 1, 2, and 5. Once Sphero reaches the carousel animal, it should be able to retrace its steps and make it back to its storage site.

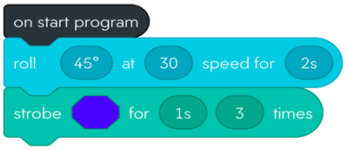


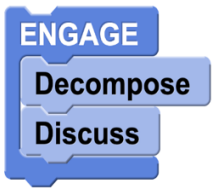
ENGAGE
Decompose
Discuss

Decompose

This is the code to get your started.

- What does this code tell Sphero to do?
- How do you think your code will compare when you move backward versus forward? What will have to change and what can stay the same?
- What do you think you are going to have to do to move to the next animal? What parts of the code that you add will have to change from the original and what parts can be copied from the original?





Discuss

Compare different groups' codes and look for similarities and differences. Are there any pieces of code that seem really efficient?



Write/Test/Debug

Take some time in your group to write, test, and debug your code.



Review

- What did you need to know to code Sphero to move between the markers?
- How did the commands change as you moved from animal to animal?
- What commands remained the same?



Reflect

- How did the relationships between the angles help you with coding?
- What might change in the code if the carousel were bigger? Why?
- What would have to change in the code if there were more animals on the carousel?



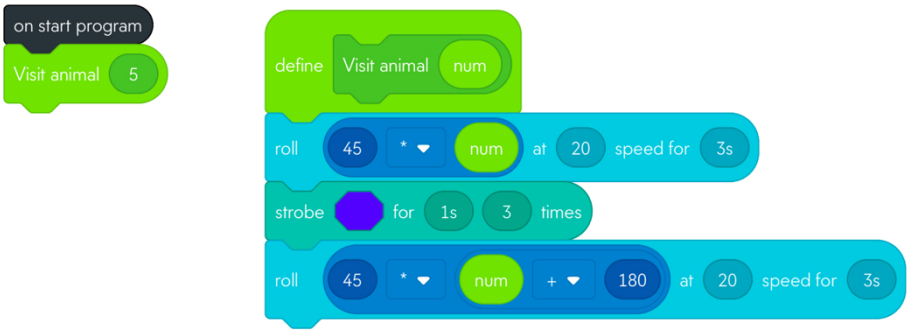
Evaluate

- Is this code efficient? Why or why not?
- What part of the code did we use over and over? Might there be a way to make the code shorter?

DEBRIEF
Review
Reflect
Evaluate

Introduction to Functions

The code below is an example of a function. Take a few minutes to look at the code and try to figure out how it makes our carousel repair program more efficient.



DEBRIEF
Review
Reflect
Evaluate

Introduction to Functions

- What should you change in this code to make Sphero go to different animals? What can remain the same? Why?
- What does “num” stand for? How is it working in the code?
- Why is it important to have the “+ 180?” What does this part of the code do?
- What does “45 * num” mean? Why does that get us to the angle we want?

Spher.Math

Lesson 11: Triangles

ENGAGE

Decompose

Discuss

Framing

Take a look at these rides from various amusement parks. What shape do you notice repeating?







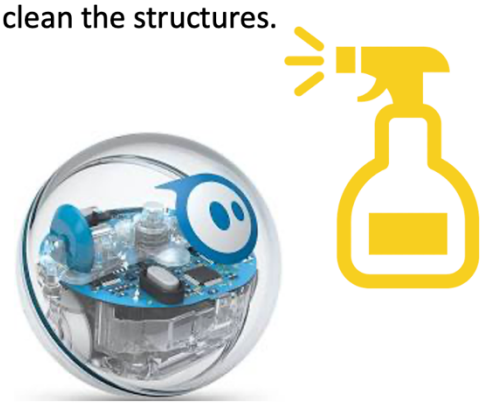


ENGAGE
Decompose
Discuss

Mission

Lots of structures use triangles because they give the structure strength and stability. Our mission today is to design a program so Sphero can travel along the triangles of the different rides in an amusement park in order to clean the structures.

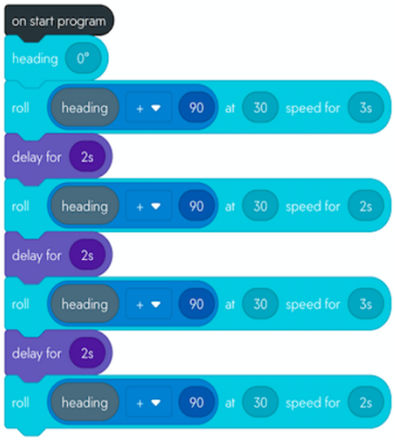
- Why would the rides need to be cleaned?
- What might be on the structure that we would need to clean it?
- Why would we want a robot to clean the rides instead of a person?

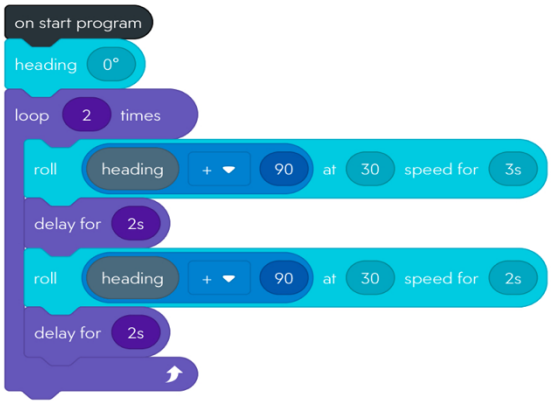


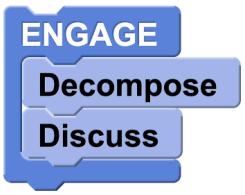
ENGAGE
Decompose
Discuss

Decompose

Think back to the Park Beautification in Lesson 11. In that challenge we used a program to draw rectangles and squares. Let's reflect on two programs from that lesson to think about drawing a triangle:







Discuss

With a partner, discuss the logic of the code.

- What do each of these programs do?
- Why do these programs draw rectangles?
- What's the difference between these two programs?



Discuss

Now think about drawing a triangle.

- What is the difference between a triangle and a rectangle?
- How is that going to change the code?
- If you are making Equilateral triangles what SHOULD each roll degree, speed and distance be?
- What blocks will you use?



Write/Test

Once you write your code, test it out by trying to make it trace one of the triangles on the floor.



Debug

- What angle were you expecting to use to trace the triangle?
  - Why is this angle not working?
- What angle does Sphero make when it moves straight ahead?
- What do we know about coding the sides of an equilateral triangle?
- Where did Sphero go compared to where you expected it to go?



Reflect

- Why do you think that the angle measurement of the actual triangle is different from the angle measurement that you coded?
- Why is angle measurement not a problem with the square?
- Did you notice any patterns in the angle measurement of the actual triangle and the angle measurements you coded?

Appendix B

Interview Questions

Student Interview Questions

1. What commands make the Sphero move? How was the speed and time helpful in determining the distance Sphero would move?
2. What was easy/hard about programming the robot to roll a given distance?
3. What commands were used multiple times?
4. What made your program successful?
5. What made your program not succeed? Why is it important for the robot to be precise? What do you have to pay attention to to get the Sphero to make accurate/precise decisions?
6. Do you prefer to program your robot to go slow or fast? Why? If the robot goes fast, does it still cover the same distance?
7. When selecting the heading for the robot, do you align the aim tail with the heading, or you choose each independently based on your position?
8. When you look at the task at hand, do you start by determining the blocks you need to program? How do you decide which blocks to use?
9. How do you identify a pattern in your program?
10. Is there a connection between the angles you enter for Sphero to change directions and the heading? Does the heading affect the distance the Sphero has to cover?

11. What strategies do you use to get your robot to complete the tasks? Do you use some codes you had used before on other tasks?

12. When your program does not accomplish the task as you expected what do you do?

13. Other than the robot and tablet, what other tools helped you accomplish your task?

14. What is the most exciting thing you have learned about programming Sphero robots?

Appendix C

Analysis Codebook

Theme	Quotes
RQ1, RQ2 Representations Embodiment	student 1: So this Sphero is supposed to go in a rectangle and end up back here, hang on hang on(moving the Sphero in a roll position on the activity board and then placing his finger on the tail light to position the Sphero on the aim position his finger stays in that position for quite some seconds student 2: so this is what you are going to do right, so you are going to look at this and tell us if anything is wrong and then you can add anything if you think anything needs to be added alright? So here's the pencil right there alright? *student 2 then run the code while student one had his finger in place on the position of the tail light for about 30 seconds.the sphero makes a rectangle and comes back to the original spot where student 1 had his finger and the student 1

	says in a satisfactory tone, "pretty good!" Me: so you're not going to add any more things to the project? student 1:wwe're gonna add much more.
	sudent 2: Alright so F, I'm going to take this one out right? (refering to a line of code on the ipad) because we're not supposed to be doing a ... student 1: no no R student 2: because we're not supposed to be doing this full thing.. student 1..no you don't understand, we have to be real trust me, trust me(and then move to the software interface ans starts coding.. when the virtual protractor pops up student 1 says" this is math, i understand this much more student 2: I am talking about the blue print, lets look at the blue print real quick, you'ree like student 1: i drew the blue print , its supposed to be a full triangle, the loudly shouts, "FULL TRIANGLE!!!" student 2: This is a rectangle(looking at the image of the rectangle that appears to have an area of discontinuity, student 2 now says)..."when does this part come in then/ (moving his finger over the area of discontinuity) does it just keep going in rectangles? (student one still making adjustments on the code, says to student 1" you are freaking...) student 2 qy=quickly cuts in.." no no no are you saying this is gonna keep going in rectangles? then what are we doing after that? student 1: And then what happened?, shouting really loudly.." there's gonna be another hall way that the Sphero goes down, aka this...(touching to the part of the drawing that connects to the rectangle student 2: alright, and then we wait for one second and the we go
RQ1. Representations	student 2: how far? student 1 so this should be about half 30 so 15, 15 is not too much hopefully, for 2 seconds, 2 seconds 15 15 second student 2: 15 is.. 15 is 90 (looking at a ruler)

Mutually supportive Mathematics and CT concepts Steps to complete the activity, pattern recognition. Mathematics Decomposition	Student 1: The sphero is supposed to pu pu pu and then blink for one second (he says this while touching the diferent spots spheros has to navigate through, in a way he is expressing the activity pattern verbally, listing the series of steps to complete the action. student 1: oh my good boy, okay, sphero be a good boy ohh go sphero(then they run the code and sphero makes the path they had intended tfor it to roll) kill hit! we hit 30 student 2: now put in the right angle, put in the right angle, start it right there student 1: you aim it (meantime while student 2 does the virtual aim, student 1 moves over to the sphero and moves one side of the activity board up.
	student 2:15 again, that was going too far (changes a few things on the application and the asks student 1 to aim it) student 1: Be a good boy sphero (then they run one more try) so one more, i don't think we have to change something. student 2: so now we need to make this long hallway(to the hallway mentioned before as shown on the blueprint) and then student 1 looks into the instructions and says, "roll backwards." student 1: so now we need to turn this hallway now (showing a turn angle on the drawing, and its not supposed to to be a right angle its actually supposed to be a 135, because it has to go backwards(now turning to the instuctions on the activity) other than 39 degress student 1: my take it means like 2 times student 2: so where is the two times component

	student1: this hallway short and this hallway is long, (pointing to parts of the drawing, meaning the length and then the width of the bigger rectangle) so it will be okay student 2: so feels like we need to make this hall way now (all of this trying to decompose the problem into parts and trying to understand the physical activity in ways that it can now be translated into a virtual format). we don't have this hallway student 1: so damn you, that hall way is right, that hall way is student 2: so where is the two times component? student 1: now we have this and this, (moving his finger on the shorter side of the rectangle, and this, touching the longer side of the rectangle) student 2: that makes sense, that makes sense student 2: we need to go faster, we need 270 you're right, 270 will go that way, moving his index finger in a curl to the left of the board student1: if 270 is essentially zero, we have to go 135 degrees, so student 2: actually we need to go 90 and more degrees, so technically we need to go what? we can go zero (inaudible) student 1: Fantastic thing (really loudly) okay so 270(moving his palm slightly to the left) is zero ; student 2 cuts in " and then 90 is" (moving his right index slightly to his right) student 1 continues sharing his thought " 270 is zero we need it to make 135 degrees" then student 2 moves to the side of the carpet uses his body to position his arguements saying "270 is going to be zero and then now we have want it to go left, 90 degrees, moving his right hand to the right with his index finger still pointing. student 1 the cuts in saying " no we don't because we have to go backwards, and angle that is not 90(these are instructions based on the activity) then student 1 reads the instructions "sphero must change direction to an angle other than 90 degrees" student 2 the says " alright, 89 degrees"

	student 2: let me show you something student 1: put 70 degrees student 2: if its wrong then Im gonna show you that I am right, put it in the right place(refering to sphero) student 2: so how long, so we have 15, so 15(inaudible) student1: no ... The goal in this activity was to make a fun house
RQ1; Symbols(Hashtag, dots functions) RQ2; Aim light	Researcher: so you have like like, one two three four five functions? student 1. yeah Researcher: Each function is represented by a hashtag. student 1: yes Student 1: The first hashtag is here; (pointing to a hashtag representation on their design) and the blue light is facing the red dot (The students in this case are taking the representations from the device, the blue tail light used to indicate the heading usually called aiming which can be used to determine the direction the robot will roll, they move this representation from the robot unto the activity they are designing. This in a bid to likely make their representations easier and less complex to manage Me: so the dot is "aim" student 1: yes Me: let me see the code, where is the main code , the big mother code. So what does this one do? (refering to the code on the ipad the students designed. student 1: that is the maze code, it deals with the maze, so when it starts here.. (pointing to parts of the activity on the mat), it goes, here, here here(pointing to a shedded area of the activity Researcher: and then, the next one? Student 1: this next one is the water park, the water park is supposed to be facing this way, (moving over and pointing to the opposite side of the activity-colored

	blue to represent water) and then it just makes a sound because we don't know what to do for that. me: so why do you use a function for that? student 1: So we don't , so we separate things student 2: so the idea is that you can choose which one they want to play, (meaning they design park themed activity and decided that functions could be used to represent each activity so park visitors can make choices as a function of the activities they decide to play...it's interesting how the students were able to integrate the concept of functions, to represent activities on an amusement park when they decide to take the idea of a park from the physical to the digital space. In the process of designing within that space they think of representations even in their language that may reflect both the idea of choice and activity within a park context where each activity has no connection with the next. student 1: so they are not confused like, so i can put this sound like the restaurant or something, it just makes it easier Researcher: and then the next one? The next function? Student 1: The next function deals with the roller coaster, it just goes ruueee Me: Okay; Student 1: this next one is the restaurant, all it says is yum yum yum , and then this last one is the one that goes student 2: to the beach Student 1: from here, (moving his fingers on the activity)...hmmm hmmm hmmm, splash splash splash Researcher: okay, so now tell me, does this mean we can put this function on any place and its gonna do something? student 1: No, because, thats why there's the instructions(picking up a piece of note paper on which they had written out color coded instructions on) and the hashtag

	marking system as you can seet he hashtags are color coded which you can see by the colors here, (showing the blue, brown ans orange color code system on the piece of paper) and then reading the color code system; brown (water park; Me: how did you come up with the idea of having colors? Student 1: so people would'nt get confused (student two breaks in saying "yeah") student 3: I think they'll also use thier imagination and italso be fun to add some colors because it could also be errmmm representative of spots and y'ahh Researcher: so why did you decide to use hashtags? how did hash tags come about? student 1: I just put them there to be easy so people don't spent too much time with how am i gonna make a way?so its just, its .... Researchere: okay, nice, lets run this
RQ1; CT Elements – Abstraction, Pattern finding. Representation	student 1: Now lets drag the beach (pulling the functions icons on their ipad) the most command Me: the most complicated command? student 1: yes me: complicated in what sense? student 1: as in.. then runs the command and a swooshing beach sound was heard from the ipad, including a roll unto the part of the activity designed to represent a beach the student then select the next function, placed it on the dot and the hashtag and then ran the code student 1: There, it now works(sphero rolls from the hashtag right unto the beach) he then continues d=saying "and then we were trying to make it go back here, roller coaster, water park comes to the maze lets turn back and then go to the, as all in one park code, but then it won't be very interactive student 3: so now we're going to go to the restaurant; she selects the function restaurant on the screen student 2 places the sphero robot on the hashtag representing the restaurant following their color code scheme and then says

RQ1 – Tailight, Virtual protractor Pattern	student 1: aim it (while student three aims the sphero using the virtual aim , student 1 moves to the activity board and gives directives as to the positioning of the aim tail light saying " the other way, no, there" making sure the tail light is alighed with the red dot the had made on the activity to represent the aim, or direction for heading the sphero will move. They in these instances were moing virtual representations both from the application icons and the robot into the physical space. student 3 then shows two other functions that do not require any aiming saying " these two do not require any aiming because it is just the sound"meaning the sphero won't make any displacements hence no need for the headings. Me: so they don't require any aiming because there is just the sound? so why did'nt you make a move from the beach to the restaurant? student 1: Because that will be too hard (indicating the challenge of finding a coding pattern Me: Because it has to go in a circle? student 3: everytime it would like bounce on the wall (student 1: yeah) Researcher: I thought you said you knew how to get sphero make a circle? So you would have made like really sharp turns? The maze, the roller coaster had dots associated with the hastags. The beach and the restaurant did not have any red dots associated with the hastag because sphero only made sounds yum yum yum representing eating and swooshing beach sounds. Different forms of representations were present in this activity
Carousel repair RQ.1 – Representations Langauge and expressions, Embodiment, Symbols, Algorithm design	student 1: It’s that easy David: so can you tell me how this works? student 1: so when you create a function block, you create a define block and then you have to define the blocks that already exists, what the block does, so usually if you have to(inaudible)ermm like (inaudible) you have to get the code block so we

	won't be using these but we will like aim(inaudible) come back we have to do that with multiple blocks but this program with just one block. We use the define block and the define block defines what we are supposed to do Interviewer: I see. student 1: its pretty simple and i have to adjust the numbers (inaudible) Interviewer: so how does it know when you say visit, when you say visit animal 7 how do you know, what makes it visit animals in the position7? student 1:errmm, this is the whole thing, it says 45 degrees, the time, the number so it will know the number, what i put, if it takes on this part only this block(scrolling to a different function on the screen)or this one, up here oh we've done that one now this one, this will (inaudible) a different number Interviewer: hmmm student 1: that's what i don't know, that's what i want to see okay fantastic, let's do it then (student then grabs the sphero from the floor and then positions it on the activity. Iinterviewer: are you aiming it, yeah, make sure it is (inaudible) student 1: so Eight is supposed to be zero(stretching his right hand in front of him to make a straight line and align with the position of animal 8 on the activity student 1: that's technically what told it to do(when the robot visits the animals on his code function) Interviewer: To go to five and to go to one? David: oop goes to 7 ..wow(when the robot makes some moves in the wrong direction student 1 picks up the robot and says" I think (inaudible) Interviewer: so what would you do to change that? how if you let your partner add it Interviewer: can i show you something to answer to your question? student 1: yes

	Interviewer: so what we just did is we added the delay block twice what would happen .. are you guys ready for this i'm gonna so what would happen if we did this? student 1: so the same effect
Square RQ1. Representation - Embodiment - Abstraction - Spatial representation - Mutually supportive maths	student 1: so, lets describe the code; so the code is supposed to make our Sphero go all around this square but if it does'nt then we will make changes. This is our current code. (screen focused on the code they had copied from their teacher)'s presentation. Researcher: so what does the code do? student 1: The code is supposed to make go ...(makes sounds while moving his hand around the square student 1: everybody ready to test it out? code stage one Researcher: what are these green (pointing to function blocks on the screen?) student 1: its errr we had to create a function Researcher: What does a function do? student 1: Its supposed to do this..(student 2 explains) student 1: lets test it, lets move it. okay, ready go ughhh we forgot about that thing... we have to aim it the other way (spatial representation)no, no no, other way, other way good, no that way, good, no, that way good there (Sphero rolls in the opposite direction)... " ughhhh" goes student 1 in frustration. now backwards student 3: you should put it this way you just said that way, its supposed to be straight, moving her right arm infront of her student 1: okay so lets make it go this way, there, noo no no okay, not long enough, it should be (inaudible) its gibberish. I think its good, its doing it people its doing it. its not doing it people its not. we need to make it longer, maybe we could try

	making that 30, it was going at 20 but maybe 30 because it only went two thirds of the width; so since we got it at 30, okay lets aim it student 3: (who usually does the virtual aiming)... ready? student 1: yes,(sphero rolls on one side of the square. "perfect" says student 1. then the robot rolls in the opposite direction ughhhhh goes the team (10:02') spatial representation. student1: yes, like no wait wait lets make it move here because then if we do that, its all about where we position it right? aim it how you normally aim (10:55') oh noo no no then it will go there its supposed to go that way(moving his right arm to make a right angle of a 90 degree turn. (12:25') you have to make it go this way, jus aim it how you normally aim it(here the students are getting confused because the aim is not following the direction they intend for it to follow, suggesting a lack of understanding in the spatial positioning of the robot and the code. student one now moves to another angle of the square and suggests to his team mates saying (12:53') I know but that's why we have to start it over here, that's what i was saying student 2: oh i understand Researcher: what do you have to start over there? student 1: because it is going this way (moving his right hand to the right) so mayif its over here, then it will it go this way(13:09) its all about positioning. (then they position the robot at an angle where the taillight is aligned to a position where if the robot rolls straight
RQ2 . Features of the programming environment and blocks.	Student 1: you guys are making me feel confused (scrolling across to find a block icon according to categories) student 2: Oh there it is, you blind? student 1: ohh oh

- Programming block Shapes - color	Student 2: then why don't you just put err student 1: we can't put it there, how are we gonna do it... i can't(moving the icon for distance to fit into the slot on the block code (speak...) student 3: Go to sensors go to sensors(scrolling across to find the sensors categories) Let me see Me: Sensors? student 3: Yes, because this is the grey part Student 1: this is hard, how do we even put this in? oh oh oh what did i do? what did i do? Me: what are you trying to add? what did she say you were going to add? student 2: yeah but she can't.. ohh just put in Me: Look at the shape of that, and then she said there was a pencil here, right? Student 1: Add string, ok so add number?... student 2: no no no no! Me: Wait, What are you going to add? Student 1: We are going to add distance in a (inaudible) student 2: find the shape of that(pointing to the shape of the space into which they have to fit the nested icon in the block Researcher: how will you define the distance? Student 2: thats the distance, (pointing to the grey distance icon) Student 1: but its grey how are we going to do it? (in a fruastrated voice) then dragging out the grey distance icon she asks her friends, "are you sure? but we need something grey" .. continues to search in the blocks icon library and asking "how are we going to do it?' student 2:wait, i think we're supposed to find the shape of distance (then they open the build string blocks and examine the shapes of the blocks in that library)

	student 1: shape of the distance shape of the distance... (scrolling through different categories of the grey block library) she then says in a frustrated voice "they're just circles" student 2: yeah they was one right there (touching the place holder for the distance on the screen student 1:No no thats not it....this hard Researcher: its hard? student 1: yeah.... getting really frustrated." how do we even do this...? Researcher: wait, hold on think carefully right?I heard the teacher when she said you have to measure and when you measure, and when you measure, what is the thing you have to enter? is it the word or a height or a weight what did she say you were going to enter into that space? its a number right? your distance is measured in what?
	Researcher: So when you put here, look at this (pointing to the dit build string block) Student 1: It says add number!!! in an excited voice (screen showing shapes of different input types to add as inner blocks to the build string block Researcher: Correct Sudent 1: Then so we add add that (dragging the round shaped block to add numbers into parent build string block \\ immediately the shape fits into the block space into which a distance can be added, the rest of the team screams .. "ohhhh..oh my God, wow, wow, wow " in excitment. Student 1: wait so how do you take that part out? (pointing to the distance icon on the parent block) student 3: I think you gotta emmm take it away with this, (touching the editable virtual pencil on the block, which the gives room to edit away the existing square

	shaped block, to only leave room for the oval shape into which they can now fit in the diastance command, when that happens, they all scream in excitement at the discovery of these new options to find string in numbers and edit the shapes to hold the numbers or variable commands.
Representations Embodiment	// Student sets the "aim" with her two feet set together. // Use of sensor data to ch check the distance
	// Student sets the "aim" with her two feet set together to align on a straight line with the "aim" light. Me: why are you putting your feet together? student 2: Because she said we have to aim it at our feet student 1: If we aim it at our feet we're gonna get..(inaudible) for sphero to...(in audible)
	Framing the question. teacher discusses the activity and describes blocks needed for the task. Teacher: Now we know what we need Sphero to do what blocks will we need? We need blocks to navigate, we need blocks to wait, we need blocks to blink right? Okay, so what blocks do you think we should use? words used to program Sphero. So now we know what we need Sphero to do, what blocks will we need? We need blocks to navigate, we need blocks to wait we need blocks to blink right? So, what blocks do you think we should use? So, we might have to do some measurements, counting all the blocks. So, counting the blocks its gonna be the measurement we need some measurement to figure out the length and then the? So, you need to figure out what is the length of you need to know, we need to measure something down here, but what block will we put that measurement in? Distance right?

RQ1: Explaining the problem Langauge Representations Pattern recognition Algorithm design Decomposition	Student 1: I think we need speed and time she said that (refering to the teacher) and probably the distance and we need to find when to stop we also need blink two times student 2: blink from yellow to green student 1. This is what we need to do okay; Student 2: yellow, green (while writing in her book) Student 2: We need roll, heading, speed, Student 1: we need roll block right? Student 2: yes, Student 1: Roll block(while writing in his note book)...We need roll block what? Student 2: Roll block, speed and time, heding stop, turn yellow to green Student 1: Don't we need?, we need speed first; student 2: yeah, student 1: Speed and then heading then speed student 2: three stop, stop for three seconds student 1: Heading... i just put heading student 2: we need to find how to blink, how to...(then reding the writing off her book) speed and time, stop, blink yellow to greenstop for...(student 1 interjects) student 1: it just needs to be then put time student 2: ohh yeah, its first, heading speed and time then ermmm then stop for three seconds and then blink yellow to green...then she moves over to the activity and moves her finger over the path on the maze student 1: There's no time on it? helpher find the block of time, he can't find it. speed time , blink yellow green. student 3: (//who had been scrolling on the tablet on the sphero app to find the different blocks); This is the first one right? Student two then counts the squares on the activity sheet to get a sense of the distance to enter into the roll block, counts up to seventeen. two for six eight, ten, twelve, fourteen, sixteen, seventeen

	student1: the speed will not be seventeen student 2:lets see student 1:no. you need to stop, this is how you do it for 17 seconds
	student 3: no two seconds, 2 seconds student 1: its 3seconds, see, one two three (counting the squares on the maze path// assuming each square equals 1second) ... then four five, then six seven eight nine, then ten eleven student 2: so three is more efficient student 1: yah or four, probably 3 is more efficient student 3: lets do blink student 1: yah lets do blink first student 3: yellow light? student 1: first is yellow then blink student 3: for three seconds? three right? student 1: three seconds student 2: do stop and wait for three seconds student 3: stop and what? //working on the tablet, entering the blocks and the values into the blocks as his team mates suggest student 2: stop and wait for three seconds student 1: stop and then wait for three seconds student 1. maybe we should stop and wait for 10 seconds instead so the people have enough time to get on Researcher: So how do you determine the number of seconds you want to stop or start? student 1: because we counted the blocks Researcher: Where did you count the blocks? student 1: here(moving his fingers on the blocks on the activity sheet and counting, one, two...) actually 1,2,3,4 that's what where we think it would stop Researcher: And each of these blocks you think is 10 seconds?

RQ1. Understanding the problem, embodiment, Gestures mutually supportive mathematics Representations – Metaphors RQ1; Counting, embodiment	Researcher: How did you come about 10 seconds from here? student 2: there's 17(moving her fingers along the maze) so if we put 10s that would work student 1: So maybe we should do.. (student 3 shouts out "5s"). Researcher: Wait, my question is this right? You have these blocks, right? How do you know the number of seconds just from looking at these blocks? How do these blocks, just from counting 1,2,3, you said how many blocks were there? student 1: Four Researcher: How would you know four blocks equal to this number of seconds? student 1: because... student 2: so we need speed times time student 1: we need to figure out how much time we need in each block before we try to find out Researcher: So you need to know how much time you need in each block before you can be able to figure out the total time from one set of blocks to the other is that correct? student 1: yes Researcher: How would you know the time in each block? student 3: this is three, because this is a stop (touching the stop sign on the activity sheet) so if it is this big, the sphero (measuring with his middle finger and thumb and moving both fingers along the maze path, counting) one two , stop (on the stop sign) and then two seconds (still moving his fingers at the right turn of the maze) student 1: or maybe we should do eight seconds because there is one line of blocks...(inaudible) Researcher: You think one line is 8seconds? how do you know that? how would you know one line is this number of seconds?

	student 3: I say three because i imagine a sphero standing here(again placing his thumb and middle fingers to represent the sphero robot) and moves his fongers in that position along the maze saying "i imagine a sphero standing here and went four blocks Researcher: so , does that tell you that... how fast is the sphero going for you to be able to count three seconds student 2: sphero is not accurate..so... student 1: sphero isn't that accurateso maybe we should do one extra second? student 1: speed times time equal distance student 3: I don't need that student 3: entering the code and moving his fingers along the maze saying " so the distance.... 16:00 Researcher: don't you see all of them have right angles? students: yeah student 2: and really its the same thing over again, (moving her fingers on the maze to follw the path) "one, two.. its like repeating three times, its the same thing so that means that all right angles student 1: all right angles student 2 so why are you putting 90? oh wait, 90 degrees is a right angle
	Researcher: so what are you trying to do? student 1: So we are trying to make Sphero do this(touching the maze with her pencil) so we're going to make it roll we're tring to figure out the speedand how much heading we need to go Me: how much heading? student 1: heading is like turning me: how much turning? student 1: yeah

	student 2: like you are in a car and then you turn right me: you are in a car and then you turn right? student 2: yeah, or the other way Researcher: and how do you know when to turn or, how do you know when to tell sphero to turn? student 1: ermmm, we program it and we put the heading blocks as much times as we need it, Researcher: so where is the block you use to make turns for sphero? student 2: this, the heading(showing a heading block on the screen) student 1: it makes Sphero turn right, it can make it turn right (moving the arrows of the virtual protractor on her screen left and then right) Researcher: so you think if you put the heading in your code.. //student one then touches her screen with the pencil indicating the position to place the heading block under the speed block student 1: lets make the speed be.. what do you think the speed could be? student 3: i think it could be in the middle like not too slow and not too fast student 1: our recent speeds are like 15, 2 emm 3, 15 i mean... which should we do 15? then i put heading, (pulling the heading block unto the block canvas) saying, that will go down //and then the virtual protractor pops up where she drags the arrow to the downward position. student 1: we also needed to stop over here first, (pointing at the screen on a position withing the code)... oh we forgot roll, where's roll, and the pulls the roll block from the movements category // using her note book, she looks at the steps on the activity, saying " roll for ...

	student 1: that's how many seconds we want it to be? roll for? how many seconds? 16s. // then looking at the activity, sheet, she points with her pencil saying " so we wnat it to go that way .. no not for 16s! sixteen seconds like for 5s will be good i guess student 2: onces it goes that way, we want it to go down student 1: //writes in notebooksaying "roll, at 14s for .
RQ1 – Pattern recognition	student 1: Now we now want it to repeat that right? student 3: yeah, because if we didn't repeat it it will be kind of boring if we just stopped like that student 3:Now we just have to do this loop, one two three four student 1: yeah, four more times student 3: i have seen four but i am still thinking five student 2: you wanna go five times? student 3: not sure student 2: it will be more fun Me: so five is as a function of what? student 1: five is as a function of how many times it will do this coding in blue(touching the codes nested into the loop blocks) Me: so where's that on your board? student 3: oh thats six, thats six stops, one two three four five six
Reusing codes, pattern Recognition, Repeat loops	student 1: errr, 2days ago but we changed the... Researcher: which one did you do two days ago? student 1: this one //(pointing to a program they had built on the screen) student 2: we combined that one and that one // talking about two separate programs they had designed they reused codes from one program previously built to work on a new program

	student 1: so we combined, we made it much easier and made... Researcher: you made this one two days ago? both students: yes student 1: and this is the much easier one//pointing to a program in which they had used the loop block Researcher: why is it easier? student 2: Because it will go all over again student 1: because there is a repeat on it, it repeats everything, it keeps repeating everything that it says student 2: yes, this is a pattern student 1: show her, okay do it, aim it //students then run the code to make the robot navigate the maze Re:searcher so, can you explain to me like the code like, the first one says roll, what part of the stops are on the first code? student 1: the first part is loop until false, it makes it repeat thats this makes it repeat, loop and we put false and we put errr... Researcher: why do you say loop until false? student 1: because true it will not move and false it moves Researcher: why did you use false and true? student 2: i didn't do that me: who did it? me okay let me see your code, let me take a photo of your code
Heading, pattern recognition	student 1: we need the roll block student 2: put the roll block then the heading Researcher: what did yo determine was the task? what are you supposed to do? student2: we are supposed to get to stop 8 student 1: yeah, and we need to

	student 2: you have to put heading...(inaudible) student 1: but don't take away that block student 3: but we might need it student 2: now find the heading student 3: yeah thats not the one // looking for a block to nest into another block student 1: it si the one student 3: how do you know? student 1: this is dark blue (color of the block) we put it there i told you to put it there see? it is the same thing student2: i said heading..oh my god he does not...distance and heading are different things my dude it says heading not distance student1: it said heading Researcher: wait, let me give you a trick. when you have a task you need to first of all determine what is it you are trying to program right? and then from that you then know which blocks you need and what you have to do. student 2: but we have to do it 8 times student 1: but if we do it 8 times, its gonna. student 3: thats the same code student 1: first we need to test it out student 2: we're basically using the old code but its 8 stops now // the maze had been extended from four to eight stops. student 3: I guess we just use the same code as last time but put it four times student 2: loop three times (counting the number of stops) Researcher: how many loops do you have this time? //student 1 starts measuring the maze with a ruler on the lower end of the steplike maze so i ask the question

	Researcher: what are you trying to do? i think it is the same like the last time, its just eight stops right? then you have to use a new code?? student 1: last time it was 18 Researcher: what is the 4 times doing? its looping 4 times right? (students all answer yes..) so we have a loop block? student 3: these two are connected to each other(touching the horizontal and vertical parts of the steplike maze on the activity board) and every single one is connected me: which one is connected to which one? show me student three: touches horizontal and vertical connections of the maze with both hands saying " these two are connecte to each other and these two are connected to each other and the reste are also connected to each other" Researcher: right student 3: like french fries. Researcher: say it again. student 3: like french fries and ketchup me: french fries and ketchup? student 3: yes Researcher: okay, which one is the french fry which one is the ketchup? student 3: this is french fry (touching the rows of squares that form the horizontal part of the maze blocks), this is ketchup (touching the column of squares that form the vertical part of the maze) Researcher: okay, french fries ketchup french fries ketchup repeat repeat repeat student 3: yeah me: okay student 2: we did 90 degrees, wait isn't it zero is that 90?, oh yeah sorry, 90 , 90 // entering the codes into the program student 1: last time it was 18 speed, zero student 3: 18: lets try 18, if it that one doesn't work then we have to set it to 17

	student 2: I got zero, what do we do from here? student 1: why's 18 not on here? student 2: what? we don't do no 18!!(checking into the old programs they had written before student 1: yes we did! student 3: why do you want to slow it down? //students go back into the old code they had written before for a shorter maze student 2: okay, ths should be our old code student 1: yeah student 2: ohh lets see.. student 3: and put the same code that ...(inaudible) student 2: 3.6 ohhh oh oh oh so its 17 , one second, its basically the same thing but we have to do it more times student 1: i don't think its the same thing if we are using different stops student 2: yeah //still entering the code into the program student 3: you can student 2: okay now we need fade// scrolling through the blocks categories to find the fade block student 3: fade is right here, hit lights duhhh..its in this one, that one ///pointing to the fade block on the lights category student 2: fade from yellow to...what student 3: green!! student 1: its yellow! student 2: you heard? yellow and green //Note: In this activity, the student try to understand the maze pattern, that had increased from four to eight. tje challenge they now faced even after having done the same activity was to understand what changes were to be effected even if they have to reuse the same

Triangle – RQ1; Embodiment RQ2; Virtual Protractor	// student starts to program the triangle, uses his finger to point the direction he wants the sphero to move He then explains using his palm how he expects the robot to move, indicating the angle with the swerve of his hand //Together with the students we then use the virtual protractor and the physical protactor to undertsand the angles on the ground (2:30) // Students try to think through the turns around a triangle and how to program the robot to roll on an equilateral triangle. The robot rolls one side and then changes direction, not following the triangle. Researcher: check your heading, aim it properly. Did you measure the distances to make sure the speed and time correspond? student 1: yes //The students run their codes, the robots make one side of the triangle, and then rolls away from the other tape to the opposite side of the triangle. The students are frustrated, stock and don't know what to do Researcher: so there is something you need to fix, use your..., now if you roll straight, // when the students changed the start position on the triangle taped to the floor, they succeed in navigating two sides of the triangle, and were unable to make the sphero to navigate the third side. Faced with this challenge, student 1 starts moving his body using the virtual protractor on the ipad to find the direction. He uses his body as orientation to directionally find how the sphero will move. At this time, he's not making any kind of mathematical knowledge to find the angles. His body and the virtual protractor become the affordable tools he can rely on to map the activity on the ground, unto his body and the virtual protractor. Interestingly, he regularly stops to measure the angles with a physical protractor, probably with the hope that the protractor could provide him with some concrete knowledge or

	number to what to enter into the direction/angle. this did not work still. The sphero kept going either straight forward or not making the angle required to make an equilateral angled triangle.
RQ1: Mutually supportive Math+Sphero	Note: The students were trying to get the robot to navigate the equilateral triangle, by rolling 60 degrees for each side. They erroneously used the physical protractors, to measure the angles of the triangle and then entered those angles into their programs. That did not work as the robot kept rolling into different directions. The Sphero robots are designed to move forward (0 degrees), turn right (90 degrees) turn left (180 degrees) and go backwards, 360 degrees. Now, for the sphero to navigate the an equilateral triangle with sides that are all the same(mathematically 60 degrees for all three sides), the robot will not navigate at 60 degree angles. For the first side, the robot will chose the smallest angle between, 0, 30, 60 and 90 degrees. It will roll 30 degrees and then add 120 degrees on each right turn to make 150 degrees and 270 degrees respectively. So the angles for an equilateral triangle are 30 degrees, 150 degrees, 270 degrees. Its not immediately obvious using the virtual protractor the angles an equilateral triangle will make using the sphero.
	//student one tries to make the robot navigate the triangle but it keeps moving away from the tape on the floor. He then resorts to moving around the triangle with his hand in front of him, as if he were mapping the triangle unto his body. after this, he goes on to discuss the program with his friends. He then goes back to the triangle on the floor and tries with his hand tries to find a bearing standing on the tip of one of the angles and then moving his body and stretched palm in front of him to make the 60 degree angle. He then goes back to the program on the ipad, suggesting the student tried to embody the triangle. In most cases, we have student embody the robot in body syntonicity, or embody representations and symbols like the aim. In this case, the student tries to use his body to understand the angles in by walking

	the perimeter of the triangle and then making gestures with his hand that could represent directions the robot may travel. His body represents the triangle and his hand directionally the robot moving infront of his body.
RQ1. Explaining the problem - Mutual mathematics	//The students try to align the virtual and physical protractor to make sense of how the physical representation of the angle will map unto the virtual and consequently the actions of the robot. This wasn't the case. Researcher: 61 and then what's the next one? student 1: zero Me: so when you turn 60 you're gonna continue straight? student 2: No Me, thats's what it says(refering to the program.) its gonna continue straight so thats what it saying here so it needs to keep turing again. So what does it turn again to do? student 1: 61 Researcher: good! Did you measure that and you are sure its 61? student 2: we already measured it and it all gave 61 Researchere: Because it is equilateral right? Me: aim it properly so it goes on a traight line. student 1: So make it this way (using his hand to motion his friend the direction the aim button shouls align to) so it could turn that way (again using hand gestures to indicate directions for the aim) Researcher: What you want to do most times is you want to try your code in parts // 2:27 students sometimes Researcher: Did you measure your distances and you are sure the speed and the time gives that distance? student 1: yes
RQ2. Virtual protractor	// He the runs the program, the robot rolls one side of the triangle, and the rolls off track , away from the other end that starts the other side of the triangle

RQ1 Emdodiment	Researcher: So, did you see your sensor data says the angle is not sharp enough, so there is something you need to do // note that te virtual and phyiscal protractors provided information they used to enter the angles at 61 degrees, through measurements with a physical protractor and then adding that data through the virtual protactor. The sensor data now came in handy to provide clearer visual image of the actual direction the robot rolled on the the cartesian coordinate plane. Two spatial representations that may be confusing for the students to navigate. One virtual the other physical. The students get really confused at this point. Researcher: so there is something you need to pay attention to, use your...use your..wait, now when you roll straight, you delay and then you have to turn, right? so if you roll straight, you turn 60, 61, ...try that and see what it does Researcher: aim properly so we can be more precise so what will you turn again now to finish this one? student 1: 90? // 9:06 and then one more time he steps on the triangle tapped to the floor, moves his body, in the directions of the sides of the triangle with his left hand infront of him, holding the ipad with his right hand, looking into the virtual protractor on the ipad, and changing the directions on the virtual protractor as he walks on the tape representing the triangle. // At this point he's embodying both the robot and the activity and trying to find the right angle using a combination of his body, and the virtual protractor. After moving around his body and changing directions on the virtual protractor as he moved, student 1 then says: 9:44 student 1: 270 Researcher: try it // 270 is definitely the angle that would turn to complete the last side of the triangle, its the angle to one side of the equilateral triangle, meaning by using his body, the virtual protractor and embodying bot the robot and the triangle,

	the student was able to find the right angle using a combination of these representations. Researcher: try that lets see. //The robot rolls two sides of the triangle and then rolls to the opposite side of the third side. Researcher: see, what's the opposite of 270 now? // Its likely there is a spatial representation challenge here at this point, starting point for the robot to roll all the sides could be on a different point. At this point its not the angle but the spatial representation of the activity.

Bibliography

Abrahamson, D., Nathan, M. J., Williams-Pierce, C., Walkington, C., Ottmar, E. R., Soto, H., & Alibali, M. W. (2020). The Future of Embodied Design for Mathematics Teaching and Learning. *Frontiers in Education*, *5*(August), 1–29. https://doi.org/10.3389/feduc.2020.00147

Alibali, M.W., Bassok, M., Solomon, K.O., Syc, S.E., & Goldin-Meadow, S. (1999). Illuminating mental representations through speech and gesture. Psychological Sci- ence, 10, 327–333.

Alibali, M. W., & Nathan, M. J. (2012). Embodiment in mathematics teaching and learning: evidence from learners' and teachers' gestures. The Journal of the Learning Sciences. https://doi.org/10.1080/ 10508406.2011.611446.

Alibali, M. W., Heath, D. C., & Myers, H. J. (2001). Effects of visibility between speaker and listener on gesture production: Some gestures are meant to be seen. Journal of Memory and Language, 44, 169–188.

Alimisis, D., & Κυνηγος, X. (2009). Teacher Education on Robotics-Enhanced Constructivist Pedagogical Methods. In *Teacher Education on RoboticsEnhanced Constructivist Pedagogical Methods*. http://www.terecop.eu/en/Products1.html

Ainsworth, S. (2006). DeFT: a conceptual framework for considering learning with multiple representations. Learning and Instruction. https://doi.org/10.1016/j.learninstruc.2006.03.001.

Atkinson, J. M., & Heritage, J. (Eds.). (1984). Structures of social action. Cambridge, UK: Cambridge University Press.

Angeli, C., Xerou, E., & Nicolaou, M. (2019). Investigating K-2 students' computational thinking skills during a problem-solving activity about the water cycle using educational robotics. *16th International Conference on Cognition and Exploratory Learning in Digital Age, CELDA 2019*, (Celda), 85–91. https://doi.org/10.33965/celda2019_2019111l011

Barsalou, L. W. (1999). Perceptual symbol systems. Behavioral and Brain Sciences, 22, 577–660

Barsalou, L. W. (2008). Grounded cognition. Annu. Rev. Psychol. 59, 617645.

Barth-Cohen, L. A., Jiang, S., Shen, J., Chen, G., & Eltoukhy, M. (2018). Interpreting and Navigating Multiple Representations for Computational Thinking in a Robotics Programming Environment. *Journal for STEM Education Research*, *1*(1–2), 119–147. https://doi.org/10.1007/s41979-018-0006-2

Barr, V., & Stephenson, C. (2011). Bringing computational thinking to K-12: What is Involved and what is the role of the computer science education community? *ACM Inroads*, *2*(1), 48–54 Barth-Cohen, L. A., Jiang, S., Shen, J., Chen, G., & Eltoukhy, M. (2018). Interpreting and Navigating Multiple Representations for Computational Thinking in a Robotics Programming Environment. *Journal for STEM Education Research*, *1*(1–2), 119–147. https://doi.org/10.1007/s41979-018-0006-2

Basu, S., Sengupta, P., & Biswas, G. (In Review). A scaffolding framework to support learning in multi-agent-based simulation environments. Research in Science Education.

Basu, S., Kinnebrew, J., Dickes, A., Farris, A. V., Sengupta, P., Winger, J., & Biswas, G. (2012). A Science Learning Environment using a Computational Thinking Approach. In: Proceedings of the 20th International Conference on Computers in Education, Singapore.

Bau, D., Gray, J., Kelleher, C., Sheldon, J., & Turbak, F. (2017). Learnable programming: Blocks and beyond. *Communications of the ACM*, *60*(6), 72–80. https://doi.org/10.1145/3015455

Bers, M. U. (2005). The TangibleK Robotics Program : Applied Computational Thinking for Young Children The TangibleK Robotics Program : Applied Computational Thinking for Young Children Background of TangibleK Robotics The TangibleK Robotics Program : Applied Computational T, 1–20.

Bers, M. U. (2010). Let the Games Begin: Civic Playing on High-Tech Consoles. *Review of General Psychology*, *14*(2), 147–153. https://doi.org/10.1037/a0019490

Bers, M. U., Flannery, L., Kazakoff, E. R., & Sullivan, A. (2014). Computational thinking and tinkering: Exploration of an early childhood robotics curriculum. *Computers and Education*, *72*, Bers, M. U. (2018). Coding as a playground: Programming and computational thinking in the early classroom. New York: Routledge.

Bers, M. (2008). Blocks to robots: Learning with technology in the early childhood classroom. New York, NY: Teachers College Press.

Bers, M.U., & Horn, M.S. (2009). Tangible programming in early childhood: Revisiting developmental assumptions through new technologies: Childhood in a digital world.

Bers, M. U., & Horn, M. M. S. (2010). Tangible programming in early childhood: Revisiting developmental assumptions through new technologies. *High-Tech Tots: Childhood in a Digital World*, 1–32.

Benitti, F. B. V. (2012). Exploring the educational potential of robotics in schools: A Systematic Review. Computers & Education, 58, 978-988.

Benham-Hutchins, M. (2010). A review of "Changing Minds: Computers, Learning, and Literacy," by Andrea A. DiSessa, 2000. *Complicity: An International Journal of Complexity and Education*, *7*(2). https://doi.org/10.29173/cmplct8956

Bih, J., Weintrop, D., Walton, M., Elby, A., & Walkoe, J. (2020). Mutually Supportive Mathematics and Computational Thinking in a Fourth-Grade Classroom. *Proceedings of ICLS 2020*,

Bieda, K. N., & Nathan, M. J. (2009). Representational disfluency in algebra: Evidence from student gestures and speech. *ZDM - International Journal on Mathematics Education*, *41*(5), 637–650. https://doi.org/10.1007/s11858-009-0198-0

Blaut, J., Stea, D., Spencer, C., and Blades, M. (2003), Mapping as a Cultural and Cognitive Universal. Annals of the Association of American Geographers, 93 (1), pp.165-185.

Blikstein, P., & Wilensky, U. (2009). An atom is known by the company it keeps: A constructionist learning environment for materials science using Agent-Based Modeling. International Journal of Computers for Mathematical Learning, 14,81–119

Bredenfeld, A., Hofmann, A., & Steinbauer, G. (2010). Robotics in education initiatives in Europe-status, shortcomings and open questions. *Proceedings of SIMPAR 2010*

Workshops Intl. Conf. on SIMULATION, MODELING, and PROGRAMMING for AUTONOMOUS ROBOTS, 568–574. http://www.terecop.eu/simpar2010/TR-TWR-2010/22-TeachingRobotics.pdf

Botički, I., Kovačević, P., Pivalica, D., & Seow, P. (2018). Identifying patterns in computational thinking problem-solving in early primary education. *ICCE 2018 - 26th International Conference on Computers in Education, Main Conference Proceedings*, 470–475.

Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. 2012 Annual Meeting of the American Educational Research Association Retrieved from https://web.media.mit.edu/~kbrennan/files/

Bruner, J. S. (1966). On cognitive growth. In J. S. Bruner, R. R. Olver, & P. M. Greenfield (Eds.), Studies in cognitive growth: A collaboration at the center for cognitive studies (pp. 1–29). New York: Wiley.

Burke, Q. (2012). The markings of a new pencil: Introducing programming-as-writing in the middle school classroom. Journal of Media Literacy Education, 4(2), 121–135.

Burleson, W. S., Harlow, D. B., Nilsen, K. J., Perlin, K., Freed, N., Jensen, C. N., . . . Muldner, K. (2018). Active learning environments with robotic tangibles: children's physical and virtual spatial programming experiences. IEEE Transactions on Learning Technologies, 11(1), 96–106. doi:10.1109/TLT.2017.2724031

Bruce, C. D., & Hawes, Z. (2015). The role of 2D and 3D mental rotation in mathematics for young children: what is it? Why does it matter? And what can we do about it?

ZDM Mathematics Education, *47*(3), 331–343. https://doi.org/10.1007/s11858-014-0637-4

Burns, N. M. (1990). Emergence of the understanding of pictures as symbols in very young children. Unpublished master's thesis; University of Illinois, Urbana-Champaign, IL.

Caballero-Gonzalez, Y. A., Mũoz-Repiso, A. G. V., & García-Holgado, A. (2019). Learning computational thinking and social skills development in young children through problem-solving with educational robotics. *ACM International Conference Proceeding Series*, 19–23. https://doi.org/10.1145/3362789.3362874

Chen, G., Shen, J., Barth-Cohen, L., Jiang, S., Huang, X., & Eltoukhy, M. (2017). Assessing elementary students' computational thinking in everyday reasoning and robotics programming. *Computers and Education*, *109*, 162–175. https://doi.org/10.1016/j.compedu.2017.03.001

Ching, C. C., & Kafai, Y. (2008). Peer pedagogy: Student collaboration and reflection in a learning-through-design project. Teachers College Record, 110(12), 2601–2632.

Ching, Y. H., Hsu, Y. C., & Baldwin, S. (2018). Developing Computational Thinking with Educational Technologies for Young Learners. *TechTrends*, *62*(6), 563–573. https://doi.org/10.1007/s11528-018-0292-7

Clements, D. H., & Meredith, J. S. (1992). Research on Logo: Effects and Efficacy. *Journal of Computing in Childhood Education*, *4*, 263–290. http://eric.ed.gov/?id=EJ474809%5Cnhttp://el.media.mit.edu/logo-foundation/pubs/papers/research_logo.html

Clement, D. H., & Merriman, S. (1988). Componential developments in Logo programming and environments. In R. E. Mayer (Ed.), Teaching and learning computer programming (pp. 13–54). Hillsdale, NJ: Erlbaum

Coburn, C. E., & Penuel, W. R. (2016). Research–Practice Partnerships in Education: Outcomes, Dynamics, and Open Questions. *Educational Researcher*, *45*(1), 48–54. https://doi.org/10.3102/0013189X16631750

Constantinou, V., & Ioannou, A. (2018). Development of computational thinking skills through educational robotics. *CEUR Workshop Proceedings*, *2193*, 1–11.

Cuoco, A., & Curcio, F. (Eds.). (2001). The roles of representation in school mathematics: 2001 NCTM yearbook. Reston: NCTM.

de Freitas, E., and Sinclair, N. (2014). Mathematics and the Body: Material Entanglements in the Classroom. New York, NY: Cambridge University Press.

David Kirsh. 2013. Embodied cognition and the magical future of interaction design. ACM Transactions on Computer-Human Interaction (TOCHI). 20, 1 (2013), 1–30.

DeLoache, J.S., Kolstad, V. & Anderson, K. (1991). Physical similarity and young children's understanding of scale models. Child Development, 62, 111-126.

DeLoache J.S, & Burns, N. (1989 November). Pictures and models; paper presented at the meeting of the Psychonomic Society Atlanta.

De Souza, C. S., Garcia, A. C. B., Slaviero, C., Pinto, H., & Repenning, A. (2011). Semiotic traces of computational thinking acquisition. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture*

Notes in Bioinformatics), *6654 LNCS*, 155–170. https://doi.org/10.1007/978-3-642-21530-8_13

Dewey, J. (1929 (1984)). The later works of John Dewey 1929: The quest for certainty. SIU Press

Dickes, A. C., & Sengupta, P. (2013). Learning natural selection in 4th grade with multi-agent-based computational models. Research in Science Education, 43(3), 921–953.

diSessa, A. A. (1993). Toward an Epistemology of Physics. *Cognition and Instruction*, *10*(2–3), 105–225. https://doi.org/10.1080/07370008.1985.9649008

diSessa AA (2000) Changing minds: computers, learning, and literacy. MIT Press, Cambridge

Douglas H. Clements & Julie Sarama (2002) Teaching with computers in early childhood education: Strategies and professional development, Journal of Early Childhood Teacher Education, 23:3, 215-226, DOI: 10.1080/1090102020230305

Edwards, L. D. (1998). Embodying mathematics and science: Microworlds as representations. *Journal of Mathematical Behavior*, *17*(1), 53–78. https://doi.org/10.1016/s0732-3123(99)80061-3

Ebersbach, M., & Hagedorn, H. (2011). The role of cognitive flexibility in the spatial representation of children'sdrawings. Journal of Cognition and Development. https://doi.org/10.1080/15248372. 2011.539526

ESRI (2003), A road map for schools and libraries, URL: http://www.esri.com/industries/k-12/roadmap.html. Visited 10.02.03.

Flannery, L. P., & Bers, M. U. (2013). Let's dance the "Robot Hokey-Pokey!": Children's programming approaches and achievement throughout early cognitive development. *Journal of Research on Technology in Education*, *46*(1), 81–101. https://doi.org/10.1080/15391523.2013.107826

Fofang, J., Weintrop, D., Walton, M., Elby, A., & Walkoe, J. (2020). Mutually supportive mathematics and computational thinking in a fourth-grade classroom. *Computer-Supported Collaborative Learning Conference, CSCL*, *3*(2006), 1389–1396

Fofang, J. B., Weintrop, D., Moon, P., & Williams-pierce, C. (2019). Computational Bodies: Grounding Computational Thinking Practices in Embodied Gesture, 171–178

Freudenthal, H. (1983). Didactical Phenomenology of Mathematical Structures. Dordrecht: Kluwer Academic Publishers.

Friebroon Yesharim, M., & Ben-Ari, M. (2018). Teaching Computer Science Concepts Through Robotics to Elementary School Children. *International Journal of Computer Science Education in Schools*, *2*(3). https://doi.org/10.21585/ijcses.v2i3.30

Froebel, F. (2005). The Education ofMan (W. N. Hailmann, Trans.). New York, NY: Dover Publications. Original work published 1885

Fuson, K. C., & Briars, D. J. (1990). Using a base-ten blocks learning/teaching approach for first-and second-grade place-value and multidigit addition and subtraction. *Journal for research in Mathematics Education*, *21*(3), 180-206.

Glenberg, A. M. (1997). What memory is for. Behavioral and Brain Sciences, 20, 1–55.

Goldin-Meadow, S., Nusbaum, H., Kelly, S. D., & Wagner, S. (2001). Explaining Math: Gesturing Lightens the Load. *Psychological Science*, *12*(6), 516–522. https://doi.org/10.1111/1467-9280.00395.

Goldin, G. A. (1982). Mathematical language and problem-solving. In R. Skemp (Ed.), Understanding the symbolism of mathematics [Special issue]. Visible Language, 16, 221-238.

Goldin, G. A. (1983). Levels of language in mathematical problem-solving. In J. C. Bergeron & N. Herscovics (Eds.), Proceedings of the Fifth Annual Meeting of PME-NA (Vol. 2, pp. 112-120). Montreal: University of Montreal Faculty of Educational Sciences. Reprinted with minor editorial changes in Janvier (1987, pp. 59-65).

Goldin, G. A. (1987). Cognitive representational systems for mathematical problem-solving. In C. Janvier (Ed.), Problems of representation in the teaching and learning of mathematics (pp. 125-145). Hillsdale, NJ: Lawrence Eribaum Associates.

Goldin, G. A. (1988a). Affective representation and mathematical problem-solving. In M. J. Behr, C. B. Lacampagne, & M. M. Wheeler (Eds.), Proceedings of the Tenth Annual Meeting of PME-NA (pp. 1-7). DeKalb, IL: Northern Illinois University Department of Mathematical Sciences.

Goldin, G. A. (1988b). The development of a model for competence in mathematical problem solving based on systems of cognitive representation. In A. Borbas (Ed.), Proceedings of the Twelfth International Conference for the Psychology of Mathematics Education (PME) (Vol. 2, pp. 358-365). Veszprem, Hungary: OOK.

Goldin, G. A. (1990). Epistemology, constructivism, and discovery learning in mathematics. In R. B. Davis, C. A. Maher, & N. Noddings (Eds.), Constructivist

views on the teaching and learning of mathematics (pp. 31-47). Reston, VA: National Council of Teachers of Mathematics, Journal for Research in Mathematics Education Monograph No. 4.

Goldin, G. A. (1992a). On developing a unified model for the psychology of mathematical learning and problem-solving. Plenary talk. In W. Geeslin & K. Graham (Eds.), Proceedings of the Sixteenth PME Conference (Vol. 3, pp. 235-261). Durham, NH: University of New Hampshire Department of Mathematics.

Goldin, G. A. (1992b). Toward an assessment framework for school mathematics. In R. Lesh & S. J. Lamon (Eds.), Assessment of authentic performance in school mathematics (pp. 63-88). Washington, DC: American Association for the Advancement of Science.

Goldin, G. A., & Herscovics, N. (1991a). The conceptual-representational analysis of children's early arithmetic. In R. Underbill (Ed.), Proceedings of the Thirteenth Annual Meeting of PME-NA (Vol. 1, pp. 118-124). Blacksburg, VA: Virginia Tech Division of Curriculum and Instruction.

Goldin, G. A., & Herscovics, N. (1991b). Toward a conceptual-representational analysis of the exponential function. In F. Furinghetti (Ed.), Proceedings of the Fifteenth Annual Conference for the Psychology of Mathematics Education (PME) (Vol. 2, pp. 64-71). Genoa, Italy: Dipartimento di Matematica dell'Universita di Geneva.

Goldin, G. A. (2000). A scientific perspective on structured, task-based interviews in mathematics education research. In R. Lesh & A. E. Kelly (Eds.), Research design in mathematics and science education (pp. 309–325). Hillsdale: Erlbaum.

Goldin-Meadow, S., Nusbaum, H., Kelly, S. D., & Wagner, S. (2001). Explaining math: gesturing lightens the load. Psychological Science. https://doi.org/10.1111/1467-9280.00395.

González, Y. A. C., & Muñoz-Repiso, A. G. V. (2018). A robotics-based approach to foster programming skills and computational thinking: Pilot experience in the classroom of early childhood education. *ACM International Conference Proceeding Series*, 41–45. https://doi.org/10.1145/3284179.3284188

Greenes, C., & Findell, C. (1999). Developing students' algebraic reasoning abilities. In L. V. Stiff & F. R. Curico (Eds.), Developing mathematical reasoning in grades K-12, 1999 yearbook (pp. 127– 137). Reston: National Council of Teachers of Mathematics.

Grover, S., & Pea, R. (2013). Computational thinking in K-12: A review of the state of the field. Educational Researcher, 42(1), 38–43.

Gunderson, E. A., Ramirez, G., Beilock, S. L., & Levine, S. C. (2012). The relation between spatial skill and early number knowledge: the role of the linear number line. Developmental Psychology. https:// doi.org/10.1037/a0027433

Hague, C., & Payton, S. (2011). Digital literacy across the curriculum. Curriculum Leadership, 9(10).

Hall, R., & Stevens, R. (2015). *Approaches To Knowledge in Use*.

Hambrusch, S., Hoffmann, C., Korb, J. T., Haugan, M., & Hosking, A. L. (2009). A multidisciplinary approach towards computational thinking for science majors. *SIGCSE'09 - Proceedings of the 40th ACM Technical Symposium on Computer Science Education*, *June 2014*, 183–187. https://doi.org/10.1145/1508865.1508931

Hammer, D., Sherin, B., & Kolpakowski, T. (1991). Inventing graphing: Meta-representational expertise in children. *Journal of Mathematical Behavior*, *10*(2), 117–160.

Harwood, D., & Usher, M. (1999). Assessing progression in primary children's map drawing skills. International Research in Geographical and Environmental Education. https://doi.org/10. 1080/10382049908667613.

Haugland, S. (1992). The effect of computer software on preschool children's developmental gains. Journal of Computing in Childhood Education, 3(1), 15–30.

Hiebert, J., & Carpenter, T. P. (1992). Learning and teaching with understanding. *Handbook of research on mathematics teaching and learning: A project of the National Council of Teachers of Mathematics*, 65-97.

Horn, M. S., Crouser, R. J., & Bers, M. U. (2012). Tangible interaction and learning: The case for a hybrid approach. *Personal and Ubiquitous Computing*, *16*(4), 379–389. https://doi.org/10.1007/s00779-011-0404-2

Horn, M., & Bers, M. (2019). Tangible Computing. *The Cambridge Handbook of Computing Education Research*, 663–678. https://doi.org/10.1017/9781108654555.023

Hostetter, A. B., & Alibali, M. W. (2008). Visible embodiment: gestures as simulated action. Psychonomic Bulletin& Review. https://doi.org/ 10.3758/PBR.15.3.495Anderson, J. R., Greeno, J. G., Reder, L. M., & Simon, H. A. (1991). Thinking , and Activity, 11–13.

Holbert, N., Berland, M., & Kafai, Y. B. (n.d.). *CONSTRUCTIONISM IN CONTEXT The Art, Theory, and Practice of Learning Designs Nathan Holbert, Matthew Berland, and Yasmin B. Kafai*.

Hostetter, A. B., & Alibali, M. W. (2010). Language, gesture, action! A test of the Gesture as Simulated Action framework. *Journal of Memory and Language*, *63*(2), 245–257. https://doi.org/10.1016/j.jml.2010.04.003

Ioannidou, A., Bennett, V., Repenning, A., Koh, K. H., & Basawapatna, A. (2011). Computational Thinking Patterns. *Online Submission*, *2*, 1–15. http://www.eric.ed.gov/ERICWebPortal/recordDetail?accno=ED520742

Israel, M., Pearson, J. N., Tapia, T., Wherfel, Q. M., & Reese, G. (2015). Supporting all learners in school-wide computational thinking: A cross-case qualitative analysis. Computers & Education, 82, 263-279.

Janvier, C. (Ed.). (1987). Problems of representation in the teaching and learning of Mathematics. Hillsdale, NJ: Lawrence Eribaum Associates.

Jones, S. R. (2018). Prototype images in mathematics education: the case of the graphical representation of the definite integral. *Educational Studies in Mathematics*, *97*(3), 215–234. https://doi.org/10.1007/s10649-017-9794-z

Johri, A., & Olds, B. M. (2011). Situated engineering learning: Bridging engineering education research and the learning sciences. Journal of Engineering Education, 100(1), 151–185. Karwowski, W., Soares, M., & Stanton, N. A. (Eds.). (2011). Handbook of human factors and ergonomics in consumer product design. Boca Raton, FL: CRC Press.

Johnson, J. (2003). Children, robotics, and education. *Artificial Life and Robotics*, *7*(1), 16-21.

Jung, S. E., & Won, E. S. (2018). Systematic review of research trends in robotics education for young children. *Sustainability (Switzerland)*, *10*(4), 1–24. https://doi.org/10.3390/su10040905.

Kafai, Y., & Burke, Q. (2013). Computer programming goes back to school. Phi Delta Kappan, 95(1), 61–65.

Kafai, Y. B., & Burke, Q. (2014). Connected code: Why children need to learn programming. Cambridge, MA: MIT Press.

Kafai, Y., Fields, D. A., & Burke, Q. (2010). Entering the clubhouse: Case studies of young programmers joining the online Scratch communities. Journal of Organizational and End User Computing, 22(2), 21–35.

Kaput, J. (1983). Representation systems and mathematics. In J. C. Bergeron & N. Herscovics (Eds.), Proceedings of the Fifth Annual Meeting of PME-NA (Vol. 2, pp. 57-66). Montreal: University of Montreal Faculty of Educational Sciences. Reprinted in Janvier (1987, pp. 19-26).

Kaput, J. (1985). Representation and problem solving: Methodological issues related to modeling. In E. Silver (Ed.), Teaching and learning mathematical problem solving: Multiple research perspectives (pp. 381-398). Hillsdale, NJ: Lawrence Eribaum Associates.

Kaput, J. (1987). Toward a theory of symbol use in mathematics. In C. Janvier (Ed.), Problems of representation in the teaching and learning of mathematics (pp. 159-196). Hillsdale, NJ: Lawrence Eribaum Associates.

Kaput, J. (1989). Linking representations in the symbol system of algebra. In C. Kieran & S. Wagner (Eds.), A research agenda for the teaching and learning of algebra (pp. 167-194). Reston, VA: National Council of Teachers of Mathematics, and Hillsdale, NJ: Lawrence Eribaum Associates.

Kaput, J. (1991). Notations and representations as mediators of constructive processes. In E. von Glasersfeld (Ed.), Constructivism in mathematics education (pp. 53-74). Dordrecht..

Kaput, J. (1994a). The representational roles of technology in connecting mathematics with authentic experience. In R. Biehler, R. W. Scholz, R. StraBer, & B. Winkelman (Eds.), Didactics of mathematics as a scientific discipline. Dordrecht: Kluwer.

Kita, S., Alibali, M. W., & Chu, M. (2017). How do gestures influence thinking and speaking? The gesture-for-conceptualization hypothe- sis. Psychological Review. https://doi.org/10.1037/rev0000059.

Kazakoff, E., & Bers, M. U. (2012). Programming in a robotics context in the kindergarten classroom: The impact on sequencing skills. *Journal of Educational Multimedia and Hypermedia*, *21*, 371–391. Retrieved from https://www.dropbox.com/sh/628r06wzowu1py6/4bA9m7cJ3PKazakoff et al 2012 Effect of a robotics and prog workshop on sequencing ability in early childhood-0422874368/Kazakoff et al 2012 Effect of a robotics and prog workshop on sequencing ability in early c

Kazakoff, E. R., & Bers, M. U. (2014). Put your robot in, put your robot out: Sequencing through programming robots in early childhood. *Journal of Educational Computing Research*, *50*(4), 553–573. https://doi.org/10.2190/EC.50.4.f

Kazakoff, E. R., Sullivan, A., & Bers, M. U. (2013). The Effect of a Classroom-Based Intensive Robotics and Programming Workshop on Sequencing Ability in Early Childhood. *Early Childhood Education Journal*, *41*(4), 245–255. https://doi.org/10.1007/s10643-012-0554-5

Kress, G. & Van Leeuwen, T. (1995), Reading Images. The Grammar of Visual Design. London: Routledge. Kress, G. & Van Leeuwen, T. (2001), Multimodal Discourse. The Modes and Media of Contemporary Communication. London: Arnold.

Kwon, H., & Capraro, M. M. (2021). Nurturing Problem Posing in Young Children: Using Multiple Representation within Students' Real-World Interest. *International Electronic Journal of Mathematics Education*, *16*(3), em0648. https://doi.org/10.29333/iejme/11066

Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W., Erickson, J., et al. (2011). Computational thinking for youth in practice. ACM Inroads, 2(1), 32–37

Lesh, R., Doerr, H. M., Carmona, G., & Hjalmarson, M. (2003). Beyond Constructivism. *Mathematical Thinking and Learning*, *5*(2–3), 211–233. https://doi.org/10.1080/10986065.2003.9680000

Lakoff, G. (1987). Women, fire, and dangerous things: What categories reveal about the mind.Chicago: University of Chicago Press.

Lakoff, G. (1999). Cognitive models and prototype theory. In E. Margolis & S. Laurence (Eds.), Concepts: Core readings (pp. 391–421). Cambridge: MIT Press.

Lesh, R., & Doerr, H. M. (2003). Foundations of models and modeling perspective on mathematics teaching, learning, and problem-solving. In R. Lesh & H. M. Doerr (Eds.), Beyond constructivism: Models and modeling perspectives on mathematics problem solving, learning, and teaching (pp. 3–34). Mahwah: Erlbaum.

Lesh, R., Landau, M., & Hamilton, E. (1983). Conceptual models and applied problem-solving research. In R. Lesh & M. Landau (Eds.), Acquisition of mathematical concepts and processes (pp. 263-343). New York: Academic Press.

Lesh, R., & Zawojewski, J. (2007). Problem-solving and modeling. In F. K. Lester Jr. (Ed.), Second handbook of research on mathematics teaching and learning (pp. 763–804). Charlotte: Information Age Publishing.

Learning Resources. (n.d.). Code & Go™ robot mouse activity set. https://www.learningresources.com/code-gor-robot-mouse-activity- set Accessed 9 July 2019.

Levy, S. T., & Mioduser, D. (2008). Does it "want" or "was it programmed to..."? Kindergarten children's explanations of an autonomous robot's adaptive functioning. *International Journal of Technology and Design Education*, *18*(4), 337–359. https://doi.org/10.1007/s10798-007-9032-6

Levy, S. T., & Mioduser, D. (2010). Approaching complexity through planful play: Kindergarten children's strategies in constructing an autonomous robot's behavior. *International Journal of Computers for Mathematical Learning*, *15*(1), 21–43. Https

Lin, Y., & Weintrop, D. (2021). 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*(October), 101075. https://doi.org/10.1016/j.cola.2021.101075.

Macedonia, M., & von Kriegstein, K. (2012). Gestures enhance foreign language learning. Biolinguistics, 6(3–4), 393–416.

Magana, A., Vieira, C., Penmetcha, M., & Matson, E. (2016). Computational Thinking as a Practice of Representation: A Proposed Learning and Assessment Framework. *The Journal of Computational Science Education*, *7*(1), 21–30. https://doi.org/10.22369/issn.2153-4136/7/1/3

Maloney, J., Resnick, M., Rusk, N., Silverman, B., & Eastmond, E. (2010). The scratch programming language and environment. *ACM Transactions on Computing Education*, *10*(4), 1–15. https://doi.org/10.1145/1868358.1868363

Maloney, J., Peppier, K., Kafai, Y. B., Resnick, M., & Rusk, N. (2008). Programming by choice: Urban youth learning programming with scratch. *SIGCSE'08 - Proceedings of the 39th ACM Technical Symposium on Computer Science Education*, 367–371. https://doi.org/10.1145/1352135.1352260

Merleau-Ponty, M. (1945/2005). Phenomenology of Perception (C. Smith, Trans.) (Original work published 1945). New York, NY: Routledge.

Mioduser, D., & Kuperman, A. (2020). Young Children's Representational Structures of Robots' Behaviors. *Design & Technology Education*, *25*(2), 143–159.

http://search.ebscohost.com/login.aspx?direct=true&db=bri&AN=144914633&site=ehost-live&authtype=ip,uid

Mioduser, D. (1998). Framework for the study of cognitive and curricular issues of technological problem-solving. International Journal of Technology and Design Education, 8(2), 167-184.

Mioduser, D., Levy, S.T., & Talis, V. (2009). Episodes to scripts to rules: Concrete-abstractions in kindergarten children's explanations of a robot's behavior. International Journal of Technology and Design Education, 19(1), 15–36. DOI: https://doi.org/10.1007/s10798-007-9040-6

Montessori, M. (1917). The advanced Montessori method. New York Frederick A. Stokes.

Montessori, M. (1967). The Absorbent Mind. (E. M. Standing, Trans.). (Original work published 1949). New York, NY: Holt, Rinehart, and Winston.

Moore, T. J., Brophy, S. P., Tank, K. M., Lopez, R. D., Johnston, A. C., Hynes, M. M., & Gajdzik, E. (2020). Multiple Representations in Computational Thinking Tasks: A Clinical Study of Second-Grade Students. *Journal of Science Education and Technology*, *29*(1), 19–34. https://doi.org/10.1007/s10956-020-09812-0

Nathan, M. J., Wolfgram, M., Srisurichan, R., Walkington, C., & Alibali, M. W. (2017). Threading mathematics through symbols, sketches, software, silicon, and wood: Teachers produce and maintain cohesion to support STEM integration. *Journal of Educational Research*, *110*(3), 272–293. https://doi.org/10.1080/00220671.2017.1287046

Nathan, M. J., Church, R. B., and Alibali, M. W. (2017a). “Making and breaking common ground: how teachers use gesture to foster learning in the classroom,” in Why Gesture? How the Hands Function in Speaking, Thinking and Communicating, eds R. Breckinridge Church, M. W. Alibali, and S. D. Kelly (Amsterdam: John Benjamins), 285–316. doi: 10.1075/gs.7. 14nat

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

NRC (2012). A framework for K-12 science education: Practices, crosscutting concepts, and core ideas. The National Academies Press.

NGSS Lead States (2013) Next generation science standards: for states, by states. The National Academies Press, Washington, DC

Noh, J., & Lee, J. (2020). Effects of robotics programming on the computational thinking and creativity of elementary school students. *Educational Technology Research and Development*, *68*(1), 463–484. https://doi.org/10.1007/s11423-019-09708-w

Ordnance Survey (2002), The GIS Files, URL: http://www.ordsvy.gov.uk/gis-files/ . Visited 10.02.03.

Papert, S. (1972). Teaching Children to be Mathematicians Versus Teaching About Mathematics. *International Journal of Mathematical Education in Science and Technology*, *3*(3), 249–262.

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. Basic books.

Papert, S. (1993). *The children’s machine: Rethinking school in the age of the computer*. Basic Books.

Papert, S. (1994). The children's machine: Rethinking school in the age of the computer. Basic Books

Pea, R. D., & Kurland, D. M. (1984). On the cognitive effects of learning computer programming. *New Ideas in Psychology*, *2*(2), 137–168. https://doi.org/10.1016/0732-118X(84)90018-7

Pea, R. (1983). Logo programming and problem-solving. In American Educational Research Association. Montreal, Canada.

Peppler, K. A., & Kafai, Y. (2007). From SuperGoo to Scratch: Exploring creative digital media production in informal learning. Learning, Media and Technology, 32(2), 149–166.

Polya, G. (1957). How to solve it (2nd ed.). Princeton, NJ: Princeton University Press.

Ratcliff, C. C., & Anderson, S. E. (2011). Reviving the turtle: Exploring the use of logo with students with mild disabilities. Computers in the Schools, 28(3), 241–255.

Piaget, J., Inhelder, B. (1956), The Child's Conception of Space. Translated by F.J. Langdon and J.L. Lunzer . London: Routledge and Kegan Paul.

Piaget, J., Inhelder, B., Szeminska, A. (1960), The Child's Conception of Geometry. Translated by E.A. Lunzer, London: Routledge and Kegan Paul.

Piaget, J. (1967). The Child's conception of the world. London, UK: Routledge & Kegan Paul.

Piaget, J. (1971b). lhe child's conception of movement and speed (G. E. T. Holloway & M. J. Mac- kenzie, Trans.). New York: Ballantine Books. (Original work published 1946)

Piaget, J., & Inhelder, B. (1971). Mental imagery in the child: A study of the development of imagi- nal representation (P. A. Chilton, Trans.). New York: Basic Books. (Original work published 1966)

Ping, R. M., & Goldin-Meadow, S. (2008). Hands in the air: using ungrounded iconic gestures to teach children conservation of quantity. *Developmental psychology*, *44*(5), 1277–1287. https://doi.org/10.1037/0012-1649.44.5.1277

Pirie, S. E. B., and Kieren, T. E. (1992). Creating constructivist environments and constructing creative mathematics. Educ. Stud. Math. 23, 505–528. doi: 10.1007/bf00571470.

Resnick, M. (2012). Point of View Reviving Papert ' s Dream, *52*(4), 42–46. Retrieved from www.media.mit.edu/~mres/papers/educational-technology-2012.pdf

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

Resnick, M., & Silverman, B. (2005). Some reflections on designing construction kits for kids. *Proceedings of: Interaction Design and Children 2005, IDC 2005*, *January 2005*, 117–122. https://doi.org/10.1145/1109540.1109556

Resnick, M. (2006). Computer as paint brush: Technology, play, and the creative society. In D. G. Singer, R. M. Golinkoff, &K. Hirsh-Pasek (Eds.), Play = learning: How play motivates and enhances chil- dren's cognitive and social-emotional growth (pp. 192–208). Oxford: Oxford University Press.

Resnick, M. (2007). All I really need to know (about creative thinking) I learned (by studying how children learn) in kindergarten. In pro- ceedings of the 6th ACM SIGCHI Conference on Creativity & Cognition (pp. 1-6). ACM.

Resnick, M. (2013, May 8). Learn to code, code to learn. EdSurge. Retrieved September 27, 2017, from https://www.edsurge.com/news/2013-05-08-learn-to-code-code-to-learn.

Romberg, T. A., Fennema, E., & Carpenter, T. P. (Eds.). (1993). Integrating research on the graphical representation offunctions. Mahwah: Lawrence Erlbaum Associates.

Roth, W. M., & Lawless, D. (2002). Science, Culture, and the Emergence of Language. *Science Education*, *86*(3), 368–385. https://doi.org/10.1002/sce.10008

Román-González, M., Pérez-González, J. C., & Jiménez-Fernández, C. (2017). Which cognitive abilities underlie computational thinking? Criterion validity ofthe Computational Thinking Test. Computers in Human Behavior. https://doi.org/10.1016/j.chb.2016.08.047.

Romero, E., Lopez, A., & Hernandez, O. (2012). A Pilot Study of Robotics in Elementary Education. *Latin American and Caribbean Conference for Engineering and Technology*, 3–4.

Rousseau, J.-J. (1979). Emile or on Education (A. Bloom, Trans.). (Originally published 1762). New York, NY: Perseus, Basic Books.

Sacks, H. (1995). Lectures on conversation (Vol. I, II). G. Jeff erson (Ed.). Oxford : Blackwell

Sáez López, J. M., Buceta Otero, R., & De Lara García-Cervigón, S. (2020). Introducing robotics and block programming in elementary education. *RIED. Revista Iberoamericana de Educación a Distancia*, *24*(1), 95. https://doi.org/10.5944/ried.24.1.27649

Sengupta, P., Kinnebrew, J. S., Basu, S., Biswas, G., & Clark, D. (2013). Integrating computational thinking with K-12 science education using agent-based computation: A theoretical framework. *Education and Information Technologies*, *18*(2), 351–380.

Sengupta, P., Dickes, A., & Farris, A. (2018). Toward a phenomenology of computational thinking in STEM education. *Computational Thinking in the STEM Disciplines: Foundations and Research Highlights*, 49–72. https://doi.org/10.1007/978-3-319-93566-9_

Sengupta, P. (2011). Design Principles for a Visual Programming Language to Integrate Agent-based Modeling in K-12 Science. In: Proceedings of the Eighth International Conference of Complex Systems (ICCS 2011), pp 1636–1637.

Sengupta, P., & Farris, A. V. (2012). Learning Kinematics in Elementary Grades Using Agent-based Computational Modeling: A Visual Programming Based Approach. Proceedings of the 11th International Conference on Interaction Design & Children,pp78–87.

Sengupta, P., & Wilensky, U. (2009). Learning electricity with NIELS: Thinking with electrons and thinking in levels. International Journal of Computers for Mathematical Learning, 14(1), 21–50.

Serrazina, L. (2017). “Day Number”: A Promoter Routine of Flexibility and Conceptual Understanding. *Journal of Mathematics Education*, *10*(2), 67–82. https://doi.org/10.26711/007577152790013

Settle, A., Franke, B., Hansen, R., Spaltro, F., Jurisson, C., Rennert-May, C., et al., 2012, July. Infusing computational thinking into the middle-and high-school curriculum. In: Proceedings of the 17th ACM annual conference on innovation and technology in computer science education. ACM, pp. 22–27.

Shim, J., Kwon, D., & Lee, W. (2017). The effects of a robot game environment on computer programming education for elementary school students. *IEEE Transactions on Education*, *60*(2), 164–172. https://doi.org/10.1109/TE.2016.2622227

Shokeen, E., Katirci, N., Bih Fofang, J., Simpson, A., & Williams-Pierce, C. (2020). Unpacking mathematical play within maker spaces using embodied cognition. *CHI PLAY 2020 - Extended Abstracts of the 2020 Annual Symposium on Computer-Human Interaction in Play*, 365–369. https://doi.org/10.1145/3383668.3419909

Shute, V. J., Sun, C., & Asbell-Clarke, J. (2017). Demystifying computational thinking. *Educational Research Review*, *22*(October), 142–158. https://doi.org/10.1016/j.edurev.2017.09.003

Skemp, R. R. (1976). Relational understanding and instrumental understanding. Math. Teach. 77, 20–26.

Smith, J. P., di Sessa, A. A., & Rochelle, J. (1993). Misconceptions reconceived: a constructivist analysis of knowledge in transition. Journal of the Learning Sciences, 3(2), 115–163.

Steffe, L. P., and Kieren, T. (1994). Radical constructivism and mathematics education. J. Res. Math. Educ. 25, 711–733.

Stoeckelmayr, K., Tesar, M., & Hofmann, A. (2011). Kindergarten children programming robots: a first attempt. *International Conference on Robotics in Education*, (March), 185–192.

Strawhacker, A., & Bers, M. U. (2015). "I want my robot to look for food": Comparing Kindergartner's programming comprehension using tangible, graphic, and hybrid user interfaces. *International Journal of Technology and Design Education*, *25*(3), 293–319. https://doi.org/10.1007/s10798-014-9287-7

Suh, J., & Moyer, P. S. (2007). Developing students' representational fluency using virtual and physical algebra balances. Journal of Computers in Mathematics and Science Teaching, 26,155 Retrieved from http://www.editlib.org/index.cfm?fuseaction=Reader.ViewFullText&paper_id=22799

Sullivan, A., & Bers, M. U. (2013). Gender differences in kindergarteners' robotics and programming achievement. *International Journal of Technology and Design Education*, *23*(3), 691–702. https://doi.org/10.1007/s10798-012-9210-z

Sullivan, A., & Umaschi, M. (2018). in Singapore ' s early childhood centers. *International Journal of Technology and Design Education*, *28*(2), 325–346. https://doi.org/10.1007/s10798-017-9397-0

Sullivan, A., Elkin, M., & Bers, M. U. (2015, June). KIBO robot demo: Engaging young children in programming and engineering. In Proceedings of the 14th International Conference on Interaction Design and Children (pp. 418-421). New York, NY: ACM.

Sullivan, A., & Bers, M. U. (2016). Robotics in the early childhood classroom: learning outcomes from an 8-week robotics curriculum in pre-kindergarten through second grade. *International Journal of Technology and Design Education*, *26*(1), 3–20. https://doi.org/10.1007/s10798-015-9304-5

Sullivan, F. R., & Heffernan, J. (2016). Robotic construction kits as computational manipulatives for learning in the STEM disciplines. *Journal of Research on Technology in Education*, *48*(2), 105–128. https://doi.org/10.1080/15391523.2016.1146563

Sykora, C. (2014, September 11). Computational thinking for all. Retrieved September 27, 2017, from https://www.iste.org/explore/articleDetail?articleid=152

Tobin White, and Roy Pea. 2011. Distributed by design: On the promises and pitfalls of collaborative learning with multiple representations. Journal of the Learning Sciences. 20, 3 (2011), 489– 547

Uttal, D. H., Scudder, K. V., & DeLoache, J. S. (1997). Manipulatives as symbols: A new perspective on the use of concrete objects to teach mathematics. *Journal of Applied Developmental Psychology*, *18*(1), 37–54. https://doi.org/10.1016/S0193-3973(97)90013-7

Varela, F. J., Thompson, E., and Rosch, E. (1991). The Embodied Mind: Cognitive Science and Human Experience. Cambridge, MA: M.I.T. Press.

von Glasersfeld, E. (1987). Preliminaries to any theory of representation. In C. Janvier (Ed.), Problems of representation in the teaching and learning of mathematics (pp. 215-226). Hillsdale, NJ: Lawrence Eribaum Associates.

Wai, J., Lubinski, D., & Benbow, C. P. (2009). Spatial ability for STEM domains: Aligning over 50 years of cumulative psychological knowledge solidifies its importance. Journal of Education & Psychology. https://doi.org/10.1037/a0016127.

Walton, M., Walkoe, J., Elby, A., Fofang, J., & Weintrop, D. (2020). Teachers' conceptualizations of computational and mathematical thinking. *Computer-Supported Collaborative Learning Conference, CSCL*, *4*, 2053–2060.

Wearne, D. & Hiebert, J. (1988). A cognitive approach to meaningful mathematics instruction: Testing a local theory using decimal numbers. Journalfor Research in Mathematics Education, 19,371-384

Weintrop, D., & Wilensky, U. (2015). To block or not to block, that is the question: Students' perceptions of blocks-based programming. *Proceedings of IDC 2015: The 14th International Conference on Interaction Design and Children*, 199–208. https://doi.org/10.1145/2771839.2771860

Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., & Wilensky, U. (2016). Defining Computational Thinking for Mathematics and Science Classrooms. *Journal of Science Education and Technology*, *25*(1), 127–147. https://doi.org/10.1007/s10956-015-9581-

Weintrop, D. (2019). Education block-based programming in computer science education. *Communications of the ACM*, *62*(8), 22–25. https://doi.org/10.1145/3341221

Weintrop, D., Wise Rutstein, D., Bienkowski, M., & McGee, S. (2021). Assessing computational thinking: an overview of the field. *Computer Science Education*, *31*(2), 113–116. https://doi.org/10.1080/08993408.2021.1918380

Weintrop, D., Walkoe, J., Walton, M., Bih, J., Moon, P., Elby, A., Bennett, B., & Kantzer, M. (2022). Sphero.Math. *Computational Thinking in PreK-5*, 39–46. https://doi.org/10.1145/3507951.3519286

Wilensky, U. (1991). "Abstract meditations on the concrete and concrete implications for mathematics education," in Constructionism, eds I. Harel and S. Papert (Norwood, NJ: Ablex Publishing Corporation), 193–204.

Wilensky, U., Brady, C. E., & Horn, M. S. (2014). Fostering Computational Literacy in Science Classrooms. *Commun. ACM*, *57*(8), 24–28. https://doi.org/10.1145/2633031.

Williams–Pierce, C. (2019). Designing for mathematical play: failure and feedback. Inform. Learn. Sci. 120, 589–610. doi: 10.1108/ils-03-2019-0027

Wilson, A. D., and Golonka, S. (2013). Embodied cognition is not what you think it is. Front. Psychol. 4:58. doi: 10.3389/fpsyg.2013.00058

Wexler, M., Kosslyn, S. M., & Berthoz, A. (1998). Motor processes in mental rotation. Cognition, 68, 77–94.

White, B. (1993). Tinkertools: Casual models, conceptual change, and science education. Cognition and Instruction, 10, 1-100.

Wilensky, U., & Resnick, M. (1999). Thinking in levels: A dynamic systems perspective to making sense of the world. Journal of Science Education and Technology, 8(1).

Wing, J. M. (2006). Computational Thinking, *49*(3), 33–35.

Wohlschläger, A., & Wohlschläger, A. (1998). Mental and manual rotation. Journal ofExperimental Psychology: Human Perception and Performance, 2, 397–412.

Wyeth, P. (2008). How young children learn to program with sensor, action, and logic blocks. *Journal of the Learning Sciences*, *17*(4), 517–550. https://doi.org/10.1080/10508400802395069

Yerushalmy, M., & Shternberg, B. (2001). Charting a visual course to the concept of function. In A. Cuoco & F. Curcio (Eds.), The roles ofrepresentations in school mathematics: 2001 NCTM yearbook (pp. 251–268). Reston: NCTM.

Zimmermann, W., & Cunningham, S. (Eds.). (1991). Visualization in teaching and learning mathematics. Washington DC: Mathematical Association of America.