Putting Computing on the Table: Using Physical Games to Teach Computer Science

Jennifer Parham-Mocello
Martin Erwig
parhammj@oregonstate.edu
erwig@oregonstate.edu
Oregon State University
Corvallis, Oregon, USA

Margaret Niess
Jason Weber
niessm@oregonstate.edu
webejaso@oregonstate.edu
Oregon State University
Corvallis, Oregon, USA

Madelyn Smith
Garrett Berliner
smitmad9@oregonstate.edu
berlineg@oregonstate.edu
Oregon State University
Corvallis, Oregon, USA

ABSTRACT

We describe a new introductory CS curriculum for middle schools that focuses on teaching CS concepts using the instructions and rules for playing simple, physical games. We deliberately avoid the use of technology and, in particular, programming, and we focus on games, such as tossing a coin to see who goes first and playing Tic-Tac-Toe. We report on middle-school students' understanding of basic CS concepts and their experiences with the curriculum.

After piloting the curriculum in 6th and 7th grade electives, we found that students liked the curriculum and using games, while some other students reported struggling with the technical content in the algorithm unit and vocabulary across the curriculum. Overall, students gained an understanding of abstraction and representation, and most students could define an algorithm and recognize a condition. However, they could not correctly organize the instructions of an algorithm. Our results suggest that the non-coding, game-based curriculum engaged middle school students in basic CS concepts at the middle school level, but we believe there is room for improvement in delivering technical content and vocabulary related to algorithms.

CCS CONCEPTS

• **Applied computing** → **Distance learning**; • **Social and professional topics** → **K-12 education**; **Computational thinking**.

KEYWORDS

CS education, middle school, games, computational thinking, unplugged

ACM Reference Format:
Jennifer Parham-Mocello, Martin Erwig, Margaret Niess, Jason Weber, Madelyn Smith, and Garrett Berliner. 2023. Putting Computing on the Table: Using Physical Games to Teach Computer Science. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1 (SIGCSE 2023), March 15–18, 2023, Toronto, ON, Canada.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3545945.3569883

SIGCSE '23, March 15–18, 2023, Toronto, ON, Canada.

ACM ISBN 978-1-4503-9431-4/23/03…$15.00
https://doi.org/10.1145/3545945.3569883

1 INTRODUCTION

We developed a curriculum for introducing computer science (CS) centered on identifying basic computing concepts in simple non-electronic games (see Figure 1). We define CS as a discipline studying the foundation of computing and all related concepts, and we use non-programming computational thinking activities and examples to illustrate fundamental concepts in CS, such as *abstraction*, *representation*, *algorithm*, and *computation*.

Figure 1: Computing explained through games.

One major goal of our approach is to debunk negative perceptions that CS is socially isolating, lacks creativity or fun, and is better suited for male students [8, 17]. We do this by demonstrating that learning basic concepts of CS is as fun, social, and gender-neutral as playing non-electronic games. Just as with games, Figure 1 shows that computing is divided into the static algorithm (or instructions) in the top half and the dynamic computation (or game play) in the bottom half. We believe choosing simple, physical games, such as tossing a coin to see who goes first or Tic-Tac-Toe, makes CS more widely accessible for students, teachers, and schools.

Our approach is similar to the approaches taken in CS For Fun (CS4FN), Teaching London Computing, and CTArcade [11, 13, 23], which also employ physical games to teach CS concepts, but it differs in a fundamental way. Instead of focusing on the strategy for winning games or playing against the computer, we use the instructions/rules for playing games without the use of a computer as a model to help students understand basic CS concepts before introducing them to programming (see Figure 1).

In the 2020/2021 academic year, a 6th grade mathematics teacher and an 8th grade mathematics teacher piloted the curriculum in their respective CS electives. To provide insights into student experiences with the new curriculum and understanding about CS concepts, we used student responses to exit tickets after each unit and a post-survey upon completion of the course to answer the following questions.

(1) What are students' understandings about abstraction, representation, and algorithms after learning about them in the curriculum?
(2) What do students like, dislike, learn, and struggle with in the curriculum?

2 MOTIVATION AND RELATED WORK

Playing games develops problem-solving skills and creativity, which are fundamental to computational thinking [18, 31, 32]. Thus, it is not surprising that games have a long tradition as learning tools in education, especially in the form of gamification, which is the idea of representing a learning process as playing a game [21]. While studies have shown that playing board games improves math skills in elementary school students [7] and involves computational thinking activities [5, 6, 19, 22], simply playing games does not increase one's computational thinking skills, unless guided instruction about the skills is given [24]. Our curriculum goes beyond just playing games by teaching core CS concepts using simple, physical games for explaining computation.

Several new board and card games have been invented specifically to teach computational thinking, such as RaBit EscAPE (ages 6-10), Cubetto (ages 3-6), and Crabs and Turtles (ages 8-9) [1, 29, 35], but new games face two challenges. First, some rules are not simple and create extraneous cognitive load on the learner, diverting cognitive resources from the learning of the computational concepts. Second, schools, kids, and families might not have access to the new games. We believe using simple, physical games instead broadens participation and shifts the focus to the computational concepts being taught.

The idea of using simple, existing physical games to explain computational concepts is not new [10, 12, 23], and researchers understand that playing games unsupported by an appropriate framework may be ineffective at teaching the computational concepts [24]. Researchers in the CS4FN and Teaching London Computing projects have shown that the use of games with well-developed lesson plans are effective for teaching specific computational concepts [11, 13], and Lee et al. have shown that their educational software called CTArcade enables children to articulate computational thinking patterns while playing Tic-Tac-Toe and Connect Four [23].

We did not use CTArcade, because we wanted students to play games with their peers to promote social interaction and communication, as well as practice concepts learned in one game by identifying them in other games, which would have to be first implemented in CTArcade.

CS Unplugged [3, 4] has been shown to broaden participation [9], and several studies have demonstrated that unplugged activities, such as games, puzzles, and storytelling, can be a viable alternative to traditional programming activities for teaching introductory computational skills and algorithms [3, 14, 15, 27, 30, 34]. Supporting studies have shown the positive impacts unplugged activities have on students' perspectives of, engagement in, and motivation to study CS [2, 14, 16, 25, 33]. Playing games is also fun and can be a source of motivation to engage the subject beyond the classroom, for example, at home with parents and friends. For these reasons, we also avoid the use of technology and programming for introducing CS concepts.

Our curriculum is unique by embracing the following features:

- Focus on game rules and not strategy to avoid competition;
- Connect game descriptions to CS concepts;
- Play games socially to promote communication and terminology;
- Avoid a programming language to promote algorithmic thinking without the use of technology.

3 CURRICULUM BACKGROUND

We developed the curriculum in collaboration with two middle school teachers. In the following, we briefly summarize the research-practice partnership (RPP) and the resulting curriculum.

3.1 A Research-Practice Partnership

Building on a well-established collaboration with a local dual-language immersion middle school, we developed the CS curriculum with two mathematics teachers, as well as the Assistant Principal, for 1-2 hours each month throughout the 2019/2020 school year. One teacher was a 6th grade mathematics teacher with a BS in primary education, and she was in her first year of teaching during the 2019/2020 development phase. The other teacher was an 8th grade mathematics teacher with a MS in secondary education, and he was in his sixth year of teaching during the development phase. Neither teacher had a background in CS or prior programming experience.

Before offering the electives, the researchers engaged with the practitioners in an iterative process to polish the curriculum and teachers' knowledge for delivering the CS content. First, the teachers participated in a summer workshop for 40 hours over two weeks addressing questions and teaching concepts related to the curriculum. This training was critical for the development of the teachers' CS content knowledge. We revised the curriculum using the information and feedback we gathered from the workshop. Then, the teachers used the curriculum to teach middle-school students in a 1-week, 3-hours-per-day online summer camp to help develop the teachers' technical pedagogical content knowledge (TPACK) [26] for teaching CS, which was especially important due to the shift to being online because of COVID-19. The one-on-one training activities and summer camps led to the final curriculum and the teachers' TPACK for delivering the electives in the 2020/2021 academic year.

3.2 The Child's Play Curriculum

The goal of our approach was to introduce basic CS concepts, such as *representation*, *abstraction*, *algorithm*, and *computation* using physical games and the human as the computer (see Figure 1). We wanted to eliminate the distraction of a machine and level the playing field between those who had and did not have prior programming experience.The resulting curriculum covered approximately

15 hours of instruction over 4.5 weeks in two units (see Table 1).

Table 1: Resulting Level 1 Curriculum.

Curriculum	CS Concepts	Games
Unit 1: Abstraction and Representation (1 week)	Abstraction Representation Kind of Thing/Type Thing/Value	*Tic-Tac-Toe*
Unit 2: Algorithm (3.5 weeks)	Algorithm Input/Output Placeholder/Variable Control Instruction (if/else and while/do) Condition	*Coin Toss* *Rock-Paper-Scissors* *Nim* *Tic-Tac-Toe*

Unit 1 contained four lessons motivating the concepts of representation (an entity that stands for something else) and abstraction (the process of omitting detail). It did this by correlating the categories/kinds of things (types) and the actual things (values) that were used in the game instructions to representations. Students played with using different representations in Tic-Tac-Toe to motivate the appropriate choice of a representation that was abstract enough to omit unnecessary detail but remained easily distinguishable, such Xs and Os versus pictures of team members or two different sides of a coin.

Unit 2 contained eight lessons which introduced the concept of algorithms and furthered the concept of representation. It did this by relating algorithms to game instructions and by addressing pros and cons of using different representations (or types of values) in a game, along with how the instructions/rules change with different representations. Unit 2 also introduced the idea of placeholders for values (variables) in algorithms and formal if-then-else constructs with conditions.

Each unit contained a cover sheet for the teacher with a list of the CS topics addressed, a unit description, objectives, definitions of CS terms, and an example of using the concepts. Each lesson had a lesson plan with an accompanying PowerPoint and student worksheet activity. The lesson plans provided teachers with a lesson overview, the details about the duration and purpose for different parts of the lesson, materials needed, and teacher instructions for the PowerPoint and student worksheet with possible student solutions (see Table 2).

The curriculum used stories to describe the games of Tic-Tac-Toe, tossing a coin, and Nim to motivate the concepts of representation and abstraction. For example, the game of Tic-Tac-Toe was initially represented as an island map with eight treasure chests and a story about two teams trying to be the first to recover a treasure by three teams members in different sections of the island pulling on the same rope (see top of Figure 2). Students were tasked with understanding the differences between the game and the story, as well as which aspects of the story were important for playing the game. Then, students were asked to think about changing the representations used in the game, such as using shapes instead of ropes and a line instead of a grid for the map, to motivate the need to talk about algorithms and how algorithms use representations (see bottom of Figure 2). After writing and presenting algorithms

Table 2: Example Unit 1 Lesson 3 Plan.

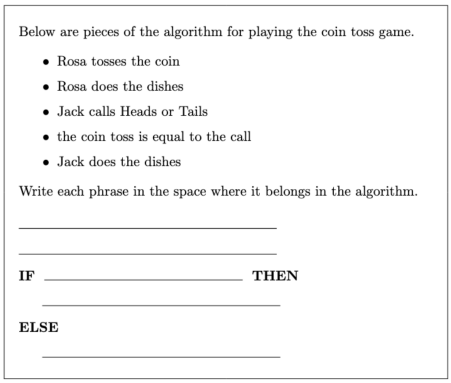
Lesson Overview		
Total Time: 36 minutes **Lesson Materials:** Lesson Overview (this document), PowerPoint slides, student worksheet, Students' homemade representations of the players for the Treasure Hunt game. **Focused Topics**: Benefits and disadvantages of representations **Summary:** Students revisit the Treasure Hunt game, this time viewing different representations of the game's components, such as using shapes instead of ropes and a single line instead of a grid for a board. Students will have to analyze the new representations to figure out how the game rules change with different representations. **Student Learning Objectives (SLOs):** Students will work on learning how to… • SLO 1: List advantages, disadvantages, and differences for various representations of game objects. • SLO 2: Understand how rules change to accommodate different representations.		
Lesson Details	**Teacher (T)**	**Student (S)**
Time: 5-10 min **Purpose:** SLO 1. Summarize ideas learned from Days 1 and 2 to refresh Ss understanding.	T has Ss place their different team characters on the classroom display as they come into the classroom (get them out of their hands!) T reminds Ss that these characters are representations for pieces of the game. Then T reminds them of some of the other representations they identified for this game, like ropes for in a row. Revisit from Day 2: Can you describe the different **representations** in this story or game and the details that were omitted by the **representation**? Today we are going to think about some of these representations and see if the game rules must change.	Ss place their team characters on the classroom display On another display are the Ss ideas for representations from yesterday such as story, island representation, ropes, symbols for ropes.

for games, we advanced students' understanding of how to formally express algorithms using the idea of Parsons Problems [28] with pieces of an algorithm jumbled and an outline for sequencing the algorithm pieces (see Figure 3).

Figure 2: Story Representation of Tic-Tac-Toe

4 EXPERIENCE IMPLEMENTATION

In this section, we describe the implementation of the electives and the data collected to answer our questions about students' understanding and experiences with our curriculum. This is an experience report from one case study, and since our population of students and teachers might not be representative of other populations, this report is not intended to be a generalizable research study.


Below are pieces of the algorithm for playing the coin toss game.

- Rosa tosses the coin
- Rosa does the dishes
- Jack calls Heads or Tails
- the coin toss is equal to the call
- Jack does the dishes

Write each phrase in the space where it belongs in the algorithm.

_____

_____

IF _____ **THEN**

_____

ELSE

_____

Figure 3: Worksheet with Algorithm as Parsons Problem

4.1 6th and 7th grade CS Electives

The curriculum was created for a 4.5-week class in the 6th grade elective wheel, which was a set of eight pre-selected electives that the majority of students rotate through. However, due to COVID-19, the school changed the elective wheel from 4.5-week classes to 9-week classes, which doubled the number of hours in the class and halved the number of offerings to four. Because 2020/2021 was the first year offering the curriculum, we also introduced the curriculum in the two 7th-grade, 18-week semester programming electives because those students were not exposed to the curriculum in the prior year. All offerings of the electives had 20-30 students.

Since school was online that year, the teachers used Zoom for class lectures and Canvas to organize all the information for the class, such as the syllabus, grades, etc. The teachers used Kahoot! [20] games as breaks from academic content for the students, and the students played online versions of the physical board games, like Boggle or Connect Four each week as a class or as a group in a breakout room.

The teachers had access to the pre-designed lesson plans, presentation slides, and student worksheets. The teachers were free to modify activities and material, add new material, or strictly keep to the lesson plans as presented. Neither teacher modified the provided curriculum before their first elective offering. However, by the end of the year, we noticed the 6th grade teacher felt more comfortable making modifications and adding content to the curriculum. We think this was because she taught the elective four times, instead of twice, and her knowledge for teaching the curriculum increased more rapidly than the 7th grade teacher's. However, since the teacher had limited but developing understanding of CS, some of the ideas for engaging students did not mesh with the designed curriculum, causing some confusion for students.

4.2 Data Collection

With IRB approval, a total of 41 students (23 identifying as male, 16 identifying as female, 1 identifying as non-binary, and 1 not available) in the 6th grade elective assented to participate in the study, and 37 students (25 identifying as male and 12 identifying as female) in the 7th grade elective assented to participate. Interestingly, only about 40% of the 6th grade students assented to be in the study; whereas, almost 80% of the 7th grade students assented.

4.2.1 Assessment Questions. At the end of both units, we provided exit tickets to assess student understanding (see Tables 3 and 4), and at the end of both electives, we provided a post survey with four open-ended questions asking students about their likes, dislikes, struggles, and learning in the elective. Not all assenting students participated in the exit tickets or post-survey. Altogether, 67 students (33 6th grade and 34 7th grade) took the Unit 1 exit ticket on abstraction and representations, and 53 students (30 6th grade and 23 7th grade) took the Unit 2 exit ticket on algorithms.

The Unit 1 exit ticket contained a combination of free-response and multiple-choice questions (see Table 3). While we intended students fill in the blank for question 5, the 6th grade teacher changed the question to a multiple-choice format in the second quarter, but the 7th grade teacher continued to keep the fill-in-the-blank format in the second semester. The Unit 2 exit ticket contained a combination of free-response and algorithm analysis questions (see Table 4). In the algorithm analysis questions 4 and 5, students were presented with a similar Parsons Problem for the coin toss algorithm they had seen in Unit 2 (see Figure 3).

Table 3: Unit 1 Open-Ended and Multiple-Choice Questions

Free Response/Fill in the blank	
1. In your own words, define abstraction. 2. In your own words, define representation. 3. Provide two examples of abstraction. 4. Provide two examples of representation. The following question was changed to multiple choice by the 6th grade teacher in the 2nd quarter, which is labeled with an 'a' to represent an alternate format. 5. In Tic-Tac-Toe, the Xs and Os _____ the players in the game.	
Multiple Choice	
Question	**Choices**
5a. In Tic-Tac-Toe, the Xs and Os _____ the players in the game.	• confirm • represent • eliminate • move
6. Which representation would be the best alternative to Xs and Os in Tic-Tac-Toe? Why?	• Using X and + • Using 0 and 1 • Using "zero" and "one" • Other
7. Which representation (Lego figures, coins, or Xs and Os) is the best choice for playing Tic-Tac-Toe? Why?	• Lego figures because they are the most fun • Xs and Os because they can be most easily distinguished • Coins because they don't omit as much detail as Xs and Os • They are all equally the best choice

Table 4: Unit 2 Open-Ended and Algorithm Questions

Free Response/Fill in the blank
1. In your own words, define algorithm. 2. What if an input is always the same for an algorithm? 3. What is the purpose of a condition in an algorithm?
Algorithm Analysis

Below are the instructions (not in the right order) for an algorithm that takes two coin tosses as input and outputs the player as a winner if the both tosses are Heads.
`1. ELSE`
`2. Get the value of the first coin`
`3. ELSE IF the second coin is equal to Heads THEN`
`4. Output Player loses!`
`5. Get the value of the second coin`
`6. IF the first coin is equal to Tails THEN`
`7. Output Player wins!`
`8. Output Player loses!`

4. List the numbers corresponding to the correct order for this algorithm. 5. List all the conditions in this algorithm.

4.2.2 Experience Questions. At the end of the 6th and 7th grade classes, students took a post-survey with four questions about their likes and dislikes about the class. Only 20 6th grade students assented to take the post-survey. We did not evaluate the experiences from the 7th grade elective, since the post-survey questions were given at the end of the elective, and the 7th grade students only engaged with the non-programming curriculum for 3-4 weeks out of their 18-week, semester course. Even though the majority of student responses focused on the programming portion of the class, there were a few students who mentioned struggling with or disliking the non-programming section of the class.

5 EXPERIENCE EVALUATION

5.1 Student Understandings

We evaluated student understandings of abstraction, representation, and algorithms using the responses to the exit tickets provided at the end of each unit in the 6th and 7th grade electives. To determine the correctness of student answers to the free-response questions, we created a rubric for coding the answers as fully, almost, or not correct. After we reached approximately 87% inter-rater reliability between two researchers on 20% of the data, we used the final rubric to code the rest of the responses to the Unit 1 and Unit 2 exit tickets.

5.1.1 What are student understandings of representation and abstraction? In the first two questions of the Unit 1 exit ticket, we asked students to define abstraction and representation in their own words. Based on what they were taught in the curriculum, we were looking for students to define abstraction as a process (or the result) of ignoring or omitting details. For representation, we wanted students to understand that a representation is an entity that stands for something else and is a form of abstraction.

Almost half of the students correctly defined abstraction in the first question, and another 42% were almost correct by giving an example of abstraction but not the definition (see Table 5). For example, one student said, "something that is more just raw shapes and not a lot of detail". When defining representation in the second question, only 39% of the students provided a correct definition, and 50% were partially correct by giving an example instead of a definition, such as saying "it is the way something is shown", as with signs. Along these same lines, when students were asked for two examples of abstraction and representation in questions 3 and 4, 84% of the students were able to provide 1-2 correct examples of both concepts, which were considered partially and fully correct.

While the performance on the first four questions did not differ based on gender or having had prior programming, the 7th grade students outperformed the 6th grade students, especially in the second question when defining representation (see Table 5). It is also interesting to note that the 6th grade students significantly improved on the first and third questions about abstraction each quarter, especially in the last two quarters.

In question 5, we asked students to fill in the blank with a word they learned in the curriculum that matched the application of the word in the sentence. Students did well on the question, and based on student responses, it did not seem that the teacher needed to change the question to a multiple choice.

Lastly, we asked students to apply their knowledge of abstraction and representation in two multiple choice questions with open areas to explain their reasoning. Students performed well on the first multiple-choice question asking them for the best alternative to Xs and Os (see Q6 in Table 5). The majority of the students recognized that a good representation for the players in Tic-Tac-Toe omits details and is easily distinguished. Students also performed really well on the second multiple-choice question asking students to recognize that Xs and Os are a good choice for Tic-Tac-Toe (see Q7 in Table 5). In both questions, over 75% of the students recognized the correct answer, but less gave a correct answer for their reasoning, which we expected at a middle school level.

Table 5: Unit 1 Percentage of Students with Correct Answers

	6th grade	7th grade	6th grade		7th grade		6th grade		7th grade	
			Female	Male	Female	Male	Prog	No Prog	Prog	No Prog
Q1	45%	53%	54%	39%	33%	64%	36%	50%	56%	44%
Q2	27%	50%	23%	22%	50%	50%	17%	36%	52%	44%
Q3	48%	65%	54%	44%	83%	55%	39%	57%	68%	56%
Q4	61%	68%	62%	61%	75%	64%	56%	64%	72%	56%
Q5	91%	91%	92%	89%	92%	91%	94%	86%	96%	78%
Q6	85%	85%	77%	89%	67%	95%	78%	93%	84%	89%
Q7	73%	79%	69%	72%	83%	77%	72%	71%	84%	67%

We considered 84-91% of 6th and 7th grade students having some understanding of what abstraction and representation meant as a success, even if students might not have had a clear understanding of how these words related to CS. One primary goal of the curriculum was to introduce students to basic CS concepts and provide them with the tools and foundation for learning CS.

5.1.2 What are student understandings of algorithms? In the first question of the second unit, we asked students to define algorithm in their own words. We were looking for students to mention one of the following facts about an algorithm: 1) An algorithm consists of instructions that are organized in some way, 2) An algorithm receives input and produces output, 3) An algorithm is a method for solving a problem, 4) An algorithm is applicable to different problem examples, or 5) An algorithm must be given by a description in a language that is understandable by a computer, which students were repeatedly told was someone or something that can understand and execute the set of instructions. The majority of the students (81%) stated one of the five facts, and another 13% gave an example of an algorithm, rather than the definition (see Table 6).

Table 6: Unit 2 Percentage of Students with Correct Answers

	6th grade	7th grade	6th grade		7th grade		6th grade		7th grade	
			Female	Male	Female	Male	Prog	No Prog	Prog	No Prog
Q1	77%	87%	75%	75%	71%	94%	88%	58%	84%	100%
Q2	53%	61%	50%	56%	57%	63%	63%	42%	58%	75%
Q3	23%	17%	42%	13%	43%	6%	19%	33%	21%	0%
Q4	27%	22%	17%	25%	0%	31%	25%	25%	21%	25%
Q5	30%	30%	25%	25%	57%	19%	25%	33%	37%	0%

In the second free-response question, we asked students to tell us what happens if the input to an algorithm was always the same, and we expected students to say something about the output always being the same. Table 6 shows that a little over half had a good understanding of the concept, and another 19% provided an example.

For example, one student said, "... if the input was always the same, the game would be boring and their [sic] would be no point to playing it." However, 25% of the students did not say either and received no credit.

The third question asked students what the purpose of a condition was, and we looked for students to say something about a condition being used to control the execution of some instructions. If they provided an example, like how if-then-else used a condition, then they were classified as being almost correct. An additional 42% gave an example of how a condition was used with a control structure, primarily with an if-then-else, than the purpose of the condition to control execution of instructions (see Table 6).

In the last two questions, we provided students with a Parsons Problem that they were to rearrange to create a correct algorithm and identify conditions (see Table 6). There were four correct orderings for the Parsons Problem: (a) 2, 5, 6, 4, 3, 7, 1, 8, (b) 2, 5, 6, 8, 3, 7, 1, 4, (c) 5, 2, 6, 4, 3, 7, 1, 8, and (d) 5, 2, 6, 8, 3, 7, 1, 4. There were two conditions: "the first coin is equal to Tails" and "the second coin is equal to Heads". Only 25% of the students correctly organized the instructions, and another quarter were close to the correct ordering. However, half of the students (51%) were not even close to correctly organizing the instructions in the Parsons Problem. Even though only 30% of the students correctly identified both conditions in the algorithm (see Table 6), another 43% identified one condition or thought the if-then-else structure was the condition. At least, only 26% of the students showed no understanding of what a condition was and did not associate it with a control structure.

Even though most students could define what an algorithm was and understand the relationship between inputs and outputs in an algorithm, they struggled with correctly organizing instructions for an algorithm that was very similar to one they saw in the curriculum. There were no significant differences in performance between students' reported gender, prior programming, or grade.

5.2 Student Experiences

In this section, we present the experiences of the 6th grade students using their responses to the post-survey questions. We categorized their qualitative responses to post-survey questions into themes. Some student responses were categorized into more than one theme, and for each survey question, we counted the number of times each theme appeared in a student response to quantitatively evaluate their overall experiences with the curriculum.

Half of the students mentioned something about games as being what they liked the most, and only one student did not like games. This may not be surprising for 11-12 year old children, but it was reassuring for the curriculum. Three students mentioned the curriculum directly, and another four students said they liked the algorithms or if-then-else part of the class.

While algorithms came up as something a few students liked in the class, algorithms were also what three students reported as liking the least in the class. Another seven students reported struggling the most with algorithms or the if-then-else in the elective, and two other students mentioned something about struggling with the game mechanics and Nim, which was when repetition/looping was introduced. Two students reported that the vocabulary was a struggle, and one student mentioned bubble sort, which was not part of the curriculum. We think this was due to the teacher introducing other concepts and curriculum to fill the extra time that was added to the elective, due to COVID-19 changes.

Just as one student mentioned struggling with something that was not part of the curriculum, three students reported learning something that was not part of the curriculum. For example, students mentioned the internet, job footprints, and Minecraft. This suggested that introducing other topics outside the curriculum might distract some students' learning the key CS concepts. On the other hand, it was encouraging to see that the majority of students reported learning about the intended concepts from the curriculum, such as algorithms, abstraction, representation, control structures, conditions, input/output, and computers.

6 CONCLUSIONS

From this experience, we learned that the teacher's technical pedagogical content knowledge (TPACK) for teaching CS, as well as the class content, contributed significantly to students' positive experiences and learning in the curriculum. We observed throughout the RPP over 2 years that middle school mathematics teachers without a background in CS could not simply use this curriculum without the proper training for understanding and teaching the content. We found that the majority of students liked the curriculum and using games as a way to explain CS concepts. Since only a few students directly mentioned not liking the non-programming part of the 7th grade elective, we believe the curriculum could be integrated into a CS programming curriculum.

Prior programming experience did not seem to influence student experiences or understandings of the material. However, the grade and quarter did impact performance on some Unit 1 questions, and as the 6th grade teacher's TPACK for teaching CS developed, the students understanding of abstraction and representation did too. It was interesting that the students' understanding of abstraction also increased as the 6th grade teacher added more activities related to abstraction, such as using the idea of selfies starting with a camera picture and then emojis.

Even though the curriculum was for 4.5 weeks/900 minutes of instruction, it seemed that teachers were able to easily adapt the curriculum to cover more or less time. However, we suggest that a teacher only add new material that employs the four features of the curriculum and reinforces the core CS concepts of the curriculum. Otherwise, a teacher might select additional material beyond student capabilities, such as with bubble sort.

As with any experience report, we recognize that the evaluation of the results were only from some students in the electives; other students might have different understandings or experiences. Therefore, the lists of understandings about key CS concepts and likes and dislikes about the curriculum are not exhaustive. However, these preliminary results provided insight into students' experiences and suggested that students learn about basic CS concepts from this curriculum. Also, these results were within the context of an online environment during COVID-19, and the experience for teachers and students might be different in-person.

ACKNOWLEDGMENTS

This work was supported by the National Science Foundation under grant #1923628.

REFERENCES

[1] P. Apostolellis, M. Stewart, C. Frisina, and D. Kafura. 2014. RaBit EscAPE: A Board Game for Computational Thinking. In *Conference on Interaction Design and Children*. 349–352.

[2] T. Bell, P. Curzon, Q. I. Cutts, V. Dagiene, and B. Haberman. 2011. Overcoming Obstacles to CS Education by Using Non-Programming Outreach Programmes. In *Int. Conf. on Informatics in Schools (LNCS 7013)*. 71–81.

[3] T. Bell, I. H. Witten, and M. Fellows. 2015. *CS Unplugged. An Enrichment and Extension Programme for Primary-Aged Students.*

[4] T. C. Bell, I. H. Witten, and M. Fellows. 1998. *Computer Science Unplugged: Off-line Activities and Games for All Ages.* Computer Science Unplugged.

[5] M. Berland and S. Duncan. 2016. Computational Thinking in the Wild: Uncovering Complex Collaborative Thinking through Gameplay. *Educational Technology* 56, 3 (2016), 29–35.

[6] M. Berland and V. R. Lee. 2011. Collaborative Strategic Board Games as a Site for Distributed Computational Thinking. *Int. Journal of Game-Based Learning* 1, 2 (2011), 65–81.

[7] S. Cavanagh. 2008. Playing Games in Class Helps Students Grasp Math. *Education Digest: Essential Readings Condensed for Quick Review* 3 (2008), 43–46.

[8] B.J. Cheryan, Drury and M Vichayapai. 2013. Enduring Influence of Stereotypical Computer Science Role Models on Women's Academic Aspirations. (2013). https://doi.org/10.1177/0361684312459328

[9] T. J. Cortina. 2015. Reaching a Broader Population of Students Through "Unplugged" Activities. *Commun. ACM* 58, 3 (2015), 25–27.

[10] CS For Fun: Queen Mary, University of London. 2011. Noughts & Crosses. http://www.cs4fn.org/programming/noughts-crosses. Accessed: 2021-01-07.

[11] CS For Fun: Queen Mary, University of London. 2011. Welcome to cs4fn : the fun side of Computer Science. http://www.cs4fn.org/. Accessed: 2021-01-07.

[12] CS For Fun: Queen Mary, University of London. 2011. Winning at Nim: computers outwitting humans. http://www.cs4fn.org/binary/nim/nim.php. Accessed: 2021-01-07.

[13] CS For Fun: Queen Mary, University of London. 2015. Teaching London Computing: A Resource Hub from CAS London & CS4FN. https://teachinglondoncomputing.org/. Accessed: 2021-01-07.

[14] Paul Curzon, Peter W. McOwan, Nicola Plant, and Laura R. Meagher. 2014. Introducing teachers to computational thinking using unplugged storytelling. *In Proceedings of the 9th Workshop in Primary and Secondary Computing Education (WiPSCE '14)*, 89–92.

[15] Q. Cutts, Q. Connor, G. Michaelson, and P. Donaldson. 2014. Code or (not code): separating formal and natural language in CS education. *Proceedings of the 9th Workshop in Primary and Secondary Computing Education*, 20–28.

[16] Q. I. Cutts, M. I. Brown, L. Kemp, and C. Matheson. 2007. Enthusing and informing potential computer science students and their teachers. In *SIGCSE Conf. on Innovation and Technology in Computer Science*. 196–200.

[17] A. Gokhale and K Machina. 2010. Online Learning Communities to Recruit and Retain Students in Information Technology Programs. (2010). https://doi.org/10.1109/ITNG.2010.259

[18] C. Harris. 2009. Meet the New School Board: Board Games Are Back–And They're Exactly What Your Curriculum Needs. *School Library Journal* 5 (2009), 24–26.

[19] N. R. Holbert and U. Wilensky. 2011. Racing games for exploring kinematics: a computational thinking approach. *7th Int.l Conf. on Games + Learning + Society*, 109–118.

[20] Kahoot! 2020. A game-based learning platform. https://kahoot.it/. Accessed: 2020-08-26.

[21] K. M. Kapp. 2012. *The Gamification of Learning and Instruction: Game-Based Methods and Strategies for Training and Education.* Pfeiffer.

[22] C. Kazimoglu, M. Kiernan, L. Bacon, and L. MacKinnon. 2012. Learning programming at the computational thinking level via digital game-play. *Procedia Computer Science* 9 (2012), 522–531.

[23] T. Y. Lee, M. L. Mauriello, J. Ahn, and B. B. Bederson. 2014. CTArcade: Computational Thinking with Games in School Age Children. *Int. Journal of Child-Computer Interaction* 2, 1 (2014), 26–33.

[24] T. Y. Lee, M. L. Mauriello, J. Ingraham, A. Sopan, J. Ahn, and B. B. Bederson. 2012. CTArcade: Learning Computational Thinking Thile Training Virtual Characters Through Game Play. In *Human Factors in Computing Systems*. 2309–2314.

[25] C. Mano, V. Allan, and D. Cooley. 2010. Effective In-Class Activities for Middle School Outreach Programs. In *Annual Conf. on Frontiers in Education*. F2E–1–F2E–6.

[26] P. Mishra and M. J. Koehler. 2006. Technological pedagogical content knowledge: A framework for teacher knowledge. *Teachers College Record* 108 (2006), 1017–1054.

[27] J. Parham-Mocello, S. Ernst, M. Erwig, E. Dominguez, and L. Shellhammer. 2019. Story Programming: Explaining Computer Science Before Coding. In *ACM SIGCSE Symp. on Computer Science Education*. 379–385.

[28] D. Parsons and P. Haden. 2006. Parson's Programming Puzzles: A Fun and Effective Learning Tool for First Programming Courses. *Proceedings of the 8th Australasian Conference on Computing Education* 52, 157–163.

[29] Primo. 2018. Cubetto: Screenless Coding Toy for Girls and Boys Aged 3-6. https://www.primotoys.com.

[30] Primo. 2020. Free beginner's guide to Coding with Kids. https://www.primotoys.com/guide-coding-for-kids-ebook/. Accessed: 2021-01-07.

[31] C. Ragatz and Z. Ragatz. 2018. Tabletop Games in a Digital World. *Parenting for High Potential* 7 (2018), 16–19.

[32] L. A. Sharp. 2012. Stealth Learning: Unexpected Learning Opportunities Through Games. *Journal of Instructional Research* 1 (2012), 42–48.

[33] R. Taub, M. Ben-Ari, and M. Armoni. 2009. The Effect of CS Unplugged on Middle-School Students' Views of CS. In *SIGCSE Conf. on Innovation and Technology in Computer Science*. 99–103.

[34] R. Thies and J. Vahrenhold. 2013. On Plugging "Unplugged" Into CS Classes. 365–370.

[35] K. Tsarava, K. Moeller, and M. Ninaus. 2018. Training Computational Thinking Through Board Games: The case of Crabs and Turtles. *Int. Journal of Serious Games* 5, 2 (2018), 25–44.