

Teaching CS Middle School Camps in a Virtual World

1st Jennifer Parham-Mocello
School of EECS
Oregon State University
Corvallis, OR, USA
parhammj@oregonstate.edu

2nd Martin Erwig
School of EECS
Oregon State University
Corvallis, OR, USA
erwig@oregonstate.edu

3rd Margaret Niess
School of EECS
Oregon State University
Corvallis, OR, USA
niessm@oregonstate.edu

Abstract—In this poster, we report our experiences with implementing two virtual computer science camps for middle school children. The camps use a two-part curriculum: One designed for 6th grade students using computer science concepts from familiar unplugged games, and the other targeted at 7th grade students using a domain-specific teaching language designed for programming board games.

We use the camps to pilot the curricular material and provide teachers with practical training to deliver the curriculum virtually. Due to the teachers' commitment to finding successful strategies for delivering the curriculum online, the unplugged and programming activities worked surprisingly well in the remote environment. Overall, we found the use of well-known, unplugged games to be effective in preparing students to think algorithmically, and students were able to successfully code small programs for simple games in a new, functional, text-based language.

I. INTRODUCTION

In this poster, we present how teachers deliver a curriculum for introducing computer science (CS) based on identifying computing concepts in well-known non-electronic games in an online setting. We think the curriculum and experience presented in this poster are especially relevant to VL/HCC's focus on virtual learning environments this year. Our curriculum is similar to CS For Fun (CS4FN), the Teaching London Computing, and CTArcade [6], [7], [14], which also employ tabletop games to teach CS concepts, but it differs in a fundamental way: Instead of focusing on the strategy for winning games or students playing against the computer, we use the rules for playing games to introduce students to CS concepts, such as representation, algorithm, and computation. More specifically, our curriculum has the following features.

- *Non-Coding First*. We deliberately avoid the teaching of a programming language in the first part of the curriculum.
- *Unplugged*. We do not use technology and embrace physical artifacts as teaching devices.
- *Games*. We employ games as the conceptual framework and metaphor for computing concepts.
- *Familiarity*. We use games that students already know.
- *Domain-Specific Language (DSL)*. We introduce students to a formal programming notation employing a newly developed DSL for describing board games.

In this study, we pilot some of our 6th and 7th grade curriculum in two one-week, online summer camps, and we use the teachers' instructional material to answer, "How do middle school teachers adopt and deliver the 6th and 7th grade CS curriculum in an impromptu, virtual summer camp environment?"

II. CURRICULUM BACKGROUND

We developed the 6th and 7th grade CS curriculum in collaboration with teachers from a local dual-language immersion middle school. The 6th grade teacher is a math teacher with a primary education degree, and the 7th grade teacher is a math teacher with a BS in Math and MS in math secondary education background. The 6th grade teacher is new to teaching, while the 7th grade teacher has been teaching for 6 years. However, neither has a CS or programming background.

A. 6th Grade Curriculum

The goal of the 6th grade curriculum is to introduce CS concepts, such as representation, abstraction, algorithm, input, output, instruction, control structures, condition, and computation, without the use of a computer. First, the concepts of representation and abstraction are motivated using a story to represent the well-known board game of Tic-Tac-Toe. Then, the concept of algorithms is introduced through the games of tossing a coin to decide who goes first and Nim.

The students are reminded throughout the curriculum that they are writing algorithms for how to play the game, instead of expressing how to win the game. This understanding helps reduce the competition of wanting to win in this curriculum. We scaffold students' understanding of how to formally express algorithms using the idea of Parsons Problems [17] with pieces of an algorithm jumbled and an outline for sequencing the algorithm pieces (see Fig. 1).

B. 7th Grade Curriculum

The goal of the 7th grade curriculum is to introduce a formal notation for algorithms within the scope of board games. To aid with learning and teaching a text-based language, we designed a *Domain Specific Teaching Language* (DSTL) called BoGL [3], [4], which is a language whose design is targeted at the particular application domain (board games in this

case) but is also shaped by the goal of moving students to a general-purpose language. BoGL is primarily a functional, text-based language syntactically similar to Haskell [9], but with a significantly simplified syntax and type system.

We support the smooth transition from algorithmic notation to a BoGL program through a process we call *BoGLization*, which shows an algorithm and the development of the BoGL program side by side, highlighting corresponding parts in both. An example is shown in Fig. 2. The partial BoGL program (shown on top) is manually created by the teacher from an algorithm (shown at the bottom) in several steps. Parts in the algorithm that have been translated in previous steps are shown in gray. The light blue background focuses on those parts in the algorithm that are translated into BoGL in the current step. This is not an automated process built into the BoGL web interface (see Fig. 3). This is a manual process that the teacher goes through with the students to build their understanding of how to go from an algorithm description to a formal programming language.

III. RELATED WORK

Playing games helps develop problem-solving skills and creativity, which are fundamental to computational thinking [8], [19], [20]. 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 [12]. While studies have shown that playing board games improves math skills in elementary school students [5] and involves computational thinking activities [1], [2], [10], [13], simply playing games does not increase one’s computational thinking skills, unless guided instruction about the skills is given [15].

Our curriculum fits into the landscape of game-based CT teaching approaches by using familiar games, instead of new ones. However, we use the same games to teach new computational concepts, unlike other curriculum that uses a different game to teach a new computational concept. Overall, our approach embraces all of the following features. (1) Focus on game rules and not strategy; (2) Connect game descriptions to CS concepts; (3) Use a DSL for expressing algorithms formally; (4) Play games to promote communication and terminology. (5) Reuse the same games as common threads; (6) Employ a text-based, functional language.

IV. RESULTS

We separated the camps into two levels to pilot each curriculum. Both teachers had access to the lesson plans, presentation slides, and student worksheets designed for the 6th and 7th grade courses, and the teachers could modify material, add new material, or strictly keep to the lesson plans.

In preparation for their school year, the teachers used Zoom and a Canvas studio site through the university to house all information for the camp, such as the schedule, introduction, and links to external documents. In recognition of middle school students’ attention spans, they divided the three hours each day into three 40 minute sessions with 10 minute breaks

and a 30-minute wrap-up discussion. Both teachers used Zoom polls as a way to keep the students engaged throughout various 5-10 minute presentations, and they used Kahoot! games as a brain break for the students [11]. Each camp started off with playing a game, and the students either played the game in pairs in breakout rooms or played as a class.

1) *Level 1 Camp*: The 6th grade teacher had exceptionally strong instructional strategies for successfully delivering the curriculum virtually. Even though we designed the curriculum to be an unplugged approach using tabletop games, the teacher used technology to remotely facilitate collaboration on algorithm design and synchronous game play. For example, the teacher used Google docs for collaborating on worksheets, Zoom breakout rooms for students to share their desktop for group work, and Padlet [16] as a way to organize and share the work that students completed. The teacher used existing websites, such as MathIsFun and Tabletopia [18], [21], to play multiplayer tabletop games remotely.

2) *Level 2 Camp*: As with level 1, the teacher had very good instructional strategies for connecting with the students. He used similar activities as the level 1 teacher for engaging the students, such as Kahoot!, grouping students, and addressing all students equally and by name. Likewise, the level 2 teacher relied heavily on the material for the structure and activities in the camp.

However, after day 1 this teacher did not use the PowerPoint slides to introduce concepts. Instead, he primarily used an electronic whiteboard for explaining concepts and kept the students in pairs using worksheets and programming activities. After students worked in groups, the teacher brought the group back together for a synthesis. The strategy of having students write the instructions and rules in a game as an algorithm to use to develop a program did not work as well as it might have if all students had been in the level 1 camp and received more directions on how to do this, which was seen in the algorithms and code developed by the different groups of students.

V. CONCLUSIONS

While the level 1 camp was only “virtually” unplugged, the use of online multiplayer tabletop games with breakout rooms for pairs of students fostered collaboration just as in a fully unplugged environment. In some ways, it forced students to communicate more precisely having to share a screen with one student entering player moves while the other student verbally explained the next move, such as first column in the first row.

We were surprised by how much prior CS and programming background students had, which proved to be more of an issue in level 1 than in level 2. While prior programming did not cause issues in the level 2 camp, teachers need to consider students’ experience with formally stating algorithms, which is learned in the level 1 camp. We were surprised by the ease at which students picked up and accepted the BoGL syntax, even though the camp was too short to develop a deep understanding of the language.

VI. ACKNOWLEDGMENT

This work has been partially supported by the National Science Foundation (NSF) through the grant DRL-1923628.

REFERENCES

- [1] M. Berland and S. Duncan. Computational Thinking in the Wild: Uncovering Complex Collaborative Thinking through Gameplay. *Educational Technology*, 56(3):29–35, 2016.
- [2] M. Berland and V. R. Lee. Collaborative Strategic Board Games as a Site for Distributed Computational Thinking. *Int. Journal of Game-Based Learning*, 1(2):65–81, 2011.
- [3] BoGL Team. A Board Game Language. <https://bogl.engr.oregonstate.edu>, 2020. Accessed: 2020-08-24.
- [4] BoGL Team. A Board Game Language – Tutorial. <https://bogl.engr.oregonstate.edu/tutorials/GettingStarted>, 2021. Accessed: 2021-06-11.
- [5] S. Cavanagh. Playing Games in Class Helps Students Grasp Math. *Education Digest: Essential Readings Condensed for Quick Review*, (3):43–46, 2008.
- [6] CS For Fun: Queen Mary, University of London. Welcome to cs4fn : the fun side of computer science. <http://www.cs4fn.org/>, 2011. Accessed: 2021-01-07.
- [7] CS For Fun: Queen Mary, University of London. Teaching london computing: A resource hub from cas london & cs4fn. <https://teachinglondoncomputing.org/>, 2015. Accessed: 2021-01-07.
- [8] C. Harris. Meet the New School Board: Board Games Are Back—And They’re Exactly What Your Curriculum Needs. *School Library Journal*, (5):24–26, 2009.
- [9] Haskell. An advanced, purely functional programming language. <https://www.haskell.org>, 2019. Accessed: 2020-08-24.
- [10] N. R. Holbert and U. Wilensky. Racing games for exploring kinematics: a computational thinking approach. pages 109–118, 2011.
- [11] Kahoot! A game-based learning platform. <https://kahoot.it/>, 2020. Accessed: 2020-08-26.
- [12] K. M. Kapp. *The Gamification of Learning and Instruction: Game-Based Methods and Strategies for Training and Education*. Pfeiffer, 2012.
- [13] C. Kazimoglu, M. Kiernan, L. Bacon, and L. MacKinnon. Learning programming at the computational thinking level via digital game-play. *Procedia Computer Science*, 9:522–531, 2012.
- [14] T. Y. Lee, M. L. Mauriello, J. Ahn, and B. B. Bederson. Ctarcade: Computational thinking with games in school age children. *Int. Journal of Child-Computer Interaction*, 2(1):26–33, 2014.
- [15] T. Y. Lee, M. L. Mauriello, J. Ingraham, A. Sopan, J. Ahn, and B. B. Bederson. CTArcade: Learning Computational Thinking Thile Training Virtual Characters Through Game Play. In *Human Factors in Computing Systems*, pages 2309–2314, 2012.
- [16] Padlet. <https://padlet.com/>. Accessed: 2021-05-05.
- [17] D. Parsons and P. Haden. Parson’s Programming Puzzles: A Fun and Effective Learning Tool for First Programming Courses. volume 52, pages 157–163, 2006.
- [18] Rod Pierce. Math is fun: Multiplayer games (html5), 2020. <https://www.mathsisfun.com/games/games-multiplayer-html5.html>.
- [19] C. Ragatz and Z. Ragatz. Tabletop Games in a Digital World. *Parenting for High Potential*, (7):16–19, 2018.
- [20] L. A. Sharp. Stealth Learning: Unexpected Learning Opportunities Through Games. *Journal of Instructional Research*, 1:42–48, 2012.
- [21] Tabletopia. Online sandbox arena for playing board games. <https://tabletopia.com/>, 2020. Accessed: 2020-08-24.

VII. APPENDICES

Lesson #2 Worksheet: Understanding Algorithms Name: _____

Below are pieces of the algorithm for playing the coin toss game to determine who does the dishes.

- Rosa tosses the coin
- Rosa does the dishes
- Jack picks Heads or Tails for the winning side
- coin toss is equal to the winning side
- Jack does the dishes

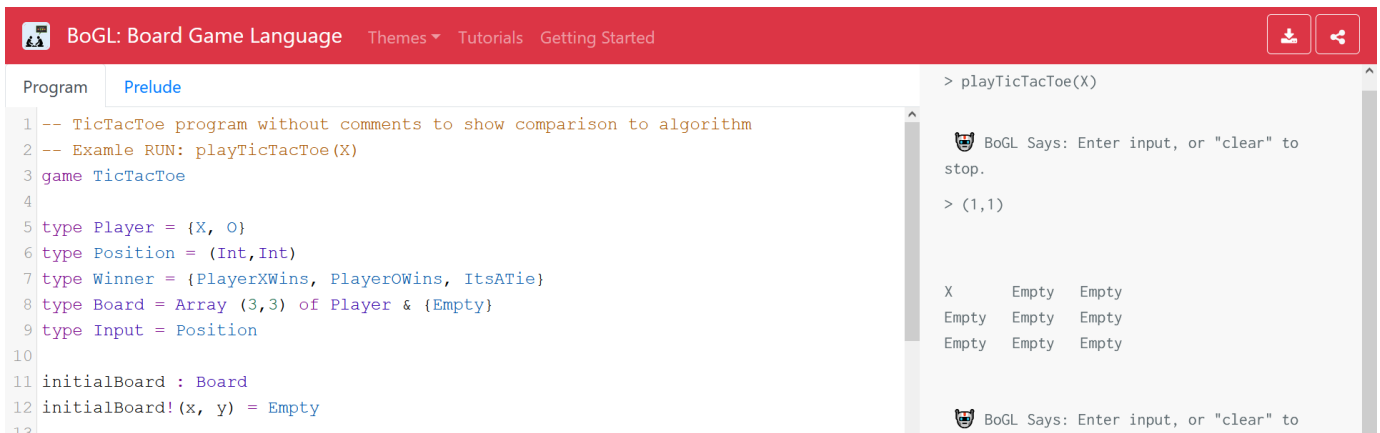
Write the phrase that belongs in each space where these pieces of the algorithm will go in the blanks.

```
_____  
_____  
IF _____ THEN  
_____  
  
ELSE  
_____
```

Fig. 1. Worksheet with an algorithm as a Parsons problem.

```
game CoinToss  
  
type Coin = {Heads, Tails}  
type Child = {Jack, Rosa}  
  
ALGORITHM Coin Toss  
Jack picks heads or tails for the winning side  
Rosa tosses the coin  
IF the coin toss is equal to the winning side THEN  
    Rosa starts  
ELSE  
    Jack starts
```

Fig. 2. BoGLization of the Coin Toss algorithm.



```
BoGL: Board Game Language Themes Tutorials Getting Started  
Program Prelude  
1 -- TicTacToe program without comments to show comparison to algorithm  
2 -- Examlle RUN: playTicTacToe(X)  
3 game TicTacToe  
4  
5 type Player = {X, O}  
6 type Position = (Int,Int)  
7 type Winner = {PlayerXWins, PlayerOWins, ItsATie}  
8 type Board = Array (3,3) of Player & {Empty}  
9 type Input = Position  
10  
11 initialBoard : Board  
12 initialBoard!(x, y) = Empty  
13  
> playTicTacToe(X)  
BoGL Says: Enter input, or "clear" to stop.  
> (1,1)  
X     Empty  Empty  
Empty Empty  Empty  
Empty Empty  Empty  
BoGL Says: Enter input, or "clear" to
```

Fig. 3. Screenshot of a Tic-Tac-Toe program in BoGL.