CTArcade: Computational Thinking with Games in School Age Children

Tak Yeon Lee
tylee@umd.edu

Matthew Louis Mauriello
mattm@cs.umd.edu

June Ahn
juneahn@umd.edu

Benjamin B. Bederson
bederson@cs.umd.edu

Human-Computer Interaction Lab,
University of Maryland, College Park, MD 20742 USA

ABSTRACT

We believe that children as young as ten can directly benefit from opportunities to engage in computational thinking. One approach to provide these opportunities is to focus on social game play. Understanding game play is common across a range of media and ages. Children can begin by solving puzzles on paper, continue on game boards, and ultimately complete their solutions on computers. Through this process, learners can be guided through increasingly complex algorithmic thinking activities that are built from their tacit knowledge and excitement about game play. This paper describes our approach to teaching computational thinking skills without traditional programming – but instead by building on children's existing game playing interest and skills. We built a system called CTArcade, with an initial game (Tic-Tac-Toe), which we evaluated with 18 children aged 10-15. The study shows that our particular approach helped young children to better draw out and articulate algorithmic thinking patterns, which were tacitly present when they played naturally on paper, but not explicitly apparent to them until they used the CTArcade interface.

Author Keywords
Computational Thinking; Learning and Games;

ACM Classification Keywords
H.5.m. Information interfaces and presentation (e.g., HCI): Miscellaneous.

General Terms
Human Factors, Design

INTRODUCTION

Researchers and educators argue that computational thinking (CT) has become a fundamental skill and should be cultivated by everyone, not just computer scientists [23]. Researchers in other disciplines, such as biology and economics, discover breakthroughs by using the same CT skills valued by computer scientists. Many everyday activities can be made more efficient when the person performing them can apply CT. However, one challenge in fostering CT is the lack of opportunities to improve one's CT skills.

Computer programming is a common way of learning CT skills, however, researchers have identified various barriers to programming [9]. Many educational programming languages and environments try to ease the process of learning to program by simplifying language syntax, using draggable graphic blocks, or by utilizing programming-by-demonstration techniques. Although these systems allow children to program without worrying about learning syntax, they still rely on the essential activity of programming that a majority of children are not familiar with. Thus, making CT skills accessible to a broader audience still remains challenging.

The use of games offers a way to engage young children in natural CT learning. Learning scientists and education researchers have found that children show a variety of CT skills while playing games [2, 7]. However, articulating those tacit skills is a very difficult task for children [13]. From these findings, we came to believe that games for learning CT should include supportive elements for building an abstract model of their game play, developing automated strategy and analyzing the performance of their strategy.

In this paper, we describe CTArcade - a gaming environment that enables children to articulate CT-related thinking patterns and conceptualize them as formal logic. We developed Tic-Tac-Toe as the first game to utilize the CTArcade environment. We report an exploratory study where we compared children playing Tic-Tac-Toe on paper to playing the game utilizing CTArcade.

The contributions of this paper are threefold: (1) We articulate an approach to developing CT skills through gameplay; (2) We provide a demonstration of the CTArcade system using a Tic-Tac-Toe game that implements a scaffolded learning structure; (3) and the exploratory user study reports findings with 18 children that show evidence that scaffolded gameplay in CTArcade helped them articulate more algorithmic thinking skills compared to playing naturally on paper.

LANGUAGES AND ENVIRONMENTS FOR CHILDREN

Papert [19] coined the term "constructionism" and observed that learning is most effective when learners construct a meaningful product. Constructionist learning theory led to initiatives such as Logo, to teach children programming in primary and secondary education. However, researchers quickly found that learning to program introduces two big barriers: the mechanics of programming and sociological factors of learners [9].

To lower the mechanical barriers to programming, researchers have contributed foundational work to create programming environments for novices. For instance, visual programming environments allow learners to create their own interactive multimedia and games by manipulating graphic (or even tangible) blocks rather than textual syntax. Examples include Lego Mindstorms [20], Scratch [21], Alice [10], HANDS [18] AgentSheets [1] and Kodu [11]. Another approach to lower mechanical barriers is Programming-by-Demonstration (PbD) [4] where learners program by showing the system what to do in specific conditions. PbD systems typically make it easy to build simple programs.

To address the sociological barriers related to learning computer programming, educational systems have been built to include a variety of sources of motivation Storytelling Alice [10] supports story creation focusing on activities teenage girls tend to enjoy. The Scratch Online Community has a library space that provides an ability to share projects with others. Robocode [17] and AlgoArena [8] feature tournament servers where competition is the motivation to learn programming. We performed a pilot study ourselves using a tournament server we built to motivate novice, undergraduate CS majors in a Java programming class. The server enabled students to create game play solutions to simple games (such as Tic-Tac-Toe, Mancala and a light-cycle simulation) and compete against other students' solutions. Based on an informal study of 150 students using that system, we confirmed the general appeal of using games to engage with computation. However, we also found some challenges that motivate the current work. For example, even a very simple game strategy requires significant expertise in programming that young children might not have.

It should be noted that while CTArcade is inspired by the various educational programming environments listed above, its goal (teaching CT skills in natural contexts) is fundamentally different. Rather than focusing on making programming easier and more fun, our goal is to teach CT skills more broadly by playing and thinking about games.

COMPUTATIONAL THINKING AND GAMES

Computational Thinking (CT) was coined by Wing [23] to describe a set of thinking patterns such as understanding problems with appropriate representation, reasoning at multiple levels of abstraction, and developing automated solutions. Google also defines four skills that are fundamental to CT; Decomposition, Pattern Recognition, Abstraction/Pattern Generalization, and Algorithm Design [6].

The positive potential for playing games as vehicles for learning CT skills has been acknowledged among learning scientists. Children playing various genres of games have shown a variety of CT patterns employed in their game play [2, 7]. A formative study [13] also found that children playing a Connect Four game demonstrate various CT-related thinking patterns. When asked to explain their strategy in abstract representation, the children showed difficulty and confusion. This led us to the thinking in this paper where we describe a platform, CTArcade, to provide scaffolding to get beyond those challenges.

CTARCADE: DESIGNING TO TEACH CT

CTArcade is a web-based educational gaming environment that extends simple games with scaffolded learning activities (see Figure 1). Our approach is to start with children's game play, and help learners conceptualize a strategy via a "Trainer" mode. In Trainer mode, children teach a virtual character how to play Tic-Tac-Toe, and then release this character to play their algorithms against other players in the simulator. The learner may then review the result of the simulator, make refinements to their strategy, and cycle through the process as many times as necessary to achieve desired goals. These iterations create a learning loop and framework for articulating CT skills. Most of the learning experience occurs within the scaffolded model that exemplifies specific CT skills, the details of which are described in the next section.

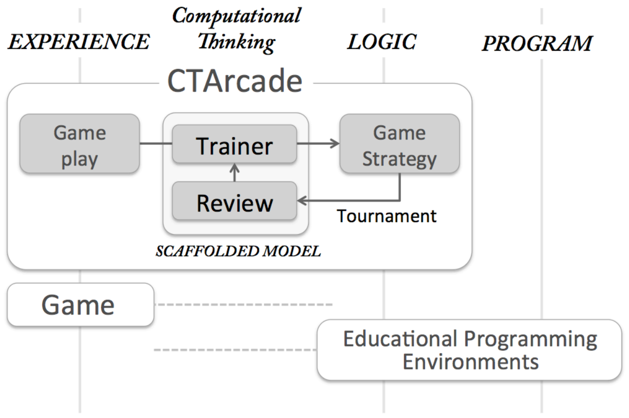


Figure 1. CTArcade framework compared with game and educational programming environments. CTArcade focuses on bridging the gap between game play (experience) and strategy (abstract logic) with a scaffolded model. The model is an instance of various CT skills.

In Figure 1, we also highlight how the CTArcade framework complements and bridges the gap between natural game play and educational programming environments. When children play games, they naturally experience situations that require CT skills, such as pattern recognition, problem decomposition, and solution building. However, these processes occur rapidly in natural context and children do not inherently conceptualize their play as abstract logic. On the other end of the spectrum, tools such as Scratch [21] help to lower the technical barrier of programming, but do not necessarily focus explicitly on general CT skills. Our goal with CTArcade is to complement these approaches by helping young children internalize CT skills while converting their game play into a formal strategy via a scaffolded model.

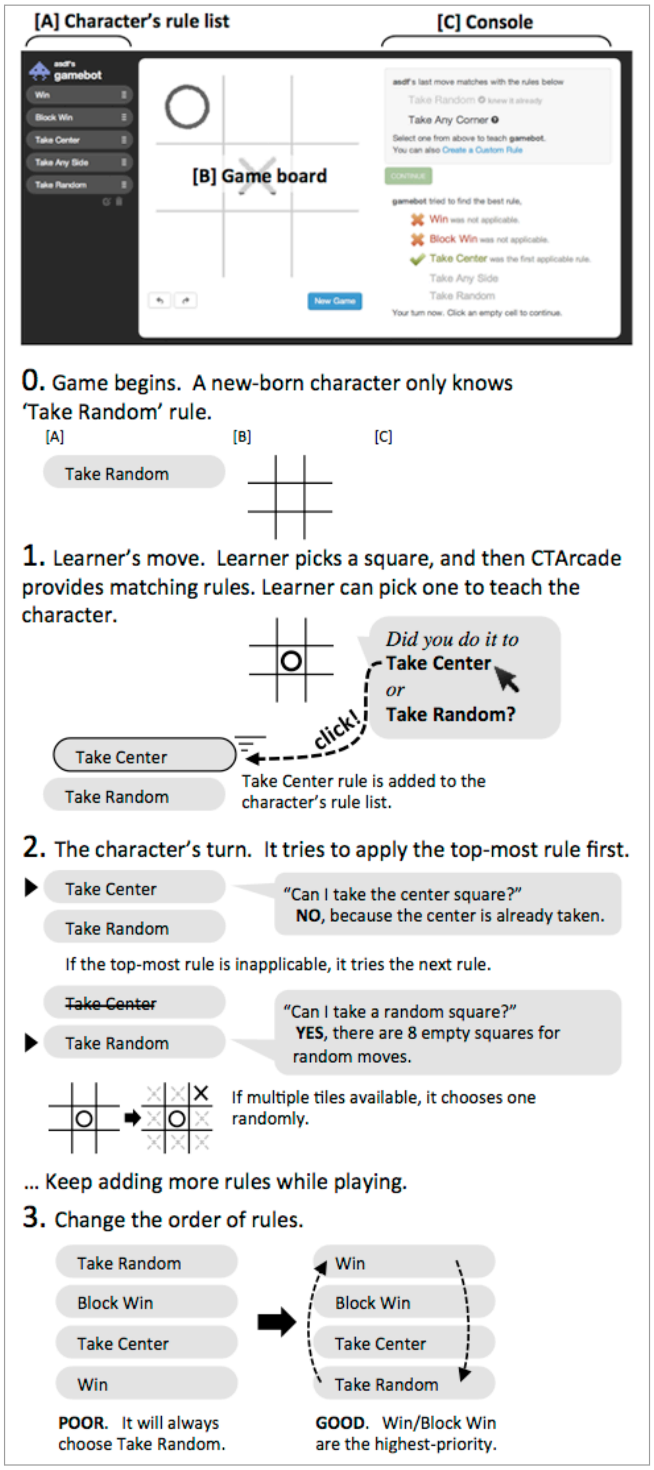
Another advantage of the CTArcade platform is that its scaffolded model is separate from the game itself. The clear division of the game from the model means that the model may be applied to other games with little modification. While many educational games exist that focus on learning, to the best of our knowledge CTArcade is the first generalizable framework applicable to many games that focuses on CT skills.

A Scaffolded Model

We designed CTArcade to scaffold the learning of CT skills via natural game play. Specific elements of the UI and environment were designed to help children articulate CT thinking patterns:

- **Templates** for game strategy help guide learners to conceptualize their game play. For example, the game strategy of Tic-Tac-Toe can be described as a prioritized list of rules, where each rule is defined by a pre/post-condition of the game board. Templates not only support different CT skills to be learned but also prevent syntactic/semantic errors. Different games may have different templates (see Future Work section for examples).
- **Learners teach virtual characters** instead of programming. We believe "teaching" provides learners with a strong motivation to conceptualize their own strategy.
- **Suggestions** are provided that infer algorithmic patterns related to the learner's game play. This is inspired by programming-by-demonstration [4] and example-driven programming synthesis [15].
- **Feedback** is provided as a direct response to a user's actions to make the refinement of strategy an easier process.
- **Visual analytic tools** facilitate iterative reflection and fine tuning of one's gameplay logic.

We chose Tic-Tac-Toe as the first game to build in the CTArcade platform. A CTArcade game consists of three modes (Trainer, Match and Review). Learners play games and teach strategy to their virtual character in the Trainer



asdf's last move matches with the rules below
Take Random knew it already
Take Any Corner
Select one from above to teach gamebot.
You can also Create a Custom Rule
gamebot tried to find the best rule,
Win was not applicable.
Block Win was not applicable.
Take Center was the first applicable rule.
Take Any Side
Take Random
Your turn now. Click an empty cell to continue.

0. Game begins. A new-born character only knows 'Take Random' rule.

1. Learner's move. Learner picks a square, and then CTArcade provides matching rules. Learner can pick one to teach the character.

Did you do it to **Take Center** *or* **Take Random?**

Take Center rule is added to the character's rule list.

2. The character's turn. It tries to apply the top-most rule first.

"Can I take the center square?" **NO**, because the center is already taken.

If the top-most rule is inapplicable, it tries the next rule.

"Can I take a random square?" **YES**, there are 8 empty squares for random moves.

If multiple tiles available, it chooses one randomly.

... Keep adding more rules while playing.

3. Change the order of rules.

POOR. It will always choose Take Random.

GOOD. Win/Block Win are the highest-priority.

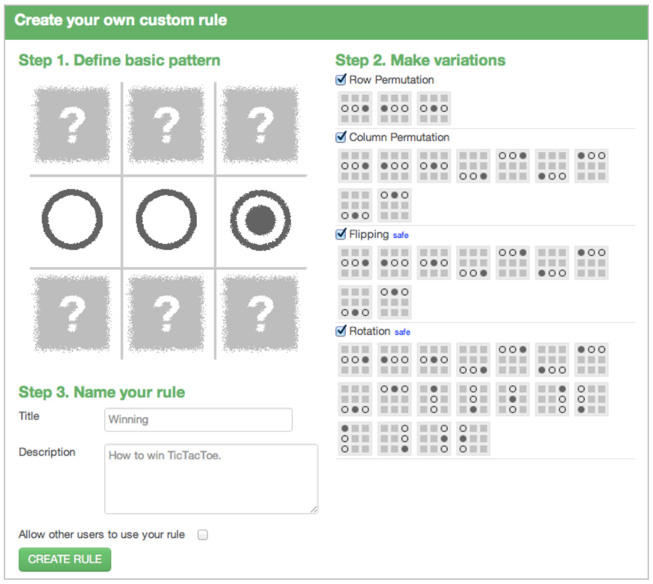
Figure 2. Trainer mode consists of; [A] Game strategy made up of rules ordered by priority; [B] game board; and [C] Console showing rules applied to learner/character's latest move.

mode. The Match mode runs an automated tournament regularly and shows the ranking of all players. In the Review mode learners can analyze a specific match result.

Trainer
The Trainer UI provides two methods of conceptualizing concrete game play examples to abstract strategy. Figure 2 describes the first (basic) method. Learners play the game and teach rules to their bot at the same time. The template we provide is a prioritized list of rules that their characters will follow, which is closely related to the skill of algorithmic thinking.

The second (advanced) method (Figure 3) is to make a new rule with the custom rule creator in two steps; 1) Defining an examplary board that shows how the rule would work; and 2) Generalizing the example by making variations . When the learner opens the custom rule creator UI, it starts with the current game board. To cover a wider range of situations, the learner might want to simplify the board by replacing irrelevant squares with question marks. The learner also needs to add at least one double-circled (next move) square. After simplifying the board, the learner can make variations with several transform operations (Row/Column Permutation, Flipping, and Rotation). Each checked operation adds board instances that the rule can be applied to while merging identical boards into one. Using this two-step method, we tried to engage learners in various opportunities to utilize their CT skills such as Abstraction/Pattern Generalization and Algorithmic Thinking.

Match
Once the game strategy is made, the learner can test it against other characters. CTArcade runs scheduled matches between all the characters every day and shows rankings. The “Match” mode is inspired by the tournament server in our pilot study as well as popular examples such as Robocode [17]. However, CTArcade lowers the barrier to engaging in match play, such that even young children, without any programming knowledge, can participate.


Create your own custom rule

Step 1. Define basic pattern

Step 2. Make variations

☑ Row Permutation

☑ Column Permutation

☑ Flipping safe

☑ Rotation safe

Step 3. Name your rule

Title: Winning

Description: How to win TicTacToe.

Allow other users to use your rule ☐

CREATE RULE

Figure 3. Custom Rule Creator. Rules are created by (Step 1) defining a basic pattern and (Step 2) applying the pattern across multiple variations. .

Review
Reflecting how well one’s strategy performs is a crucial part of the learning experience. However, the nature of multi-player games means that reviewing multiple match results requires strong Decomposition, Pattern Recognition and Generalization/Abstraction skills. CTArcade provides an integrated review process that enables learners to recognize patterns from winning and losing games. By making opportunities for identifying patterns easier, the learner is able to make generalizations about Tic-Tac-Toe strategy (see 4).

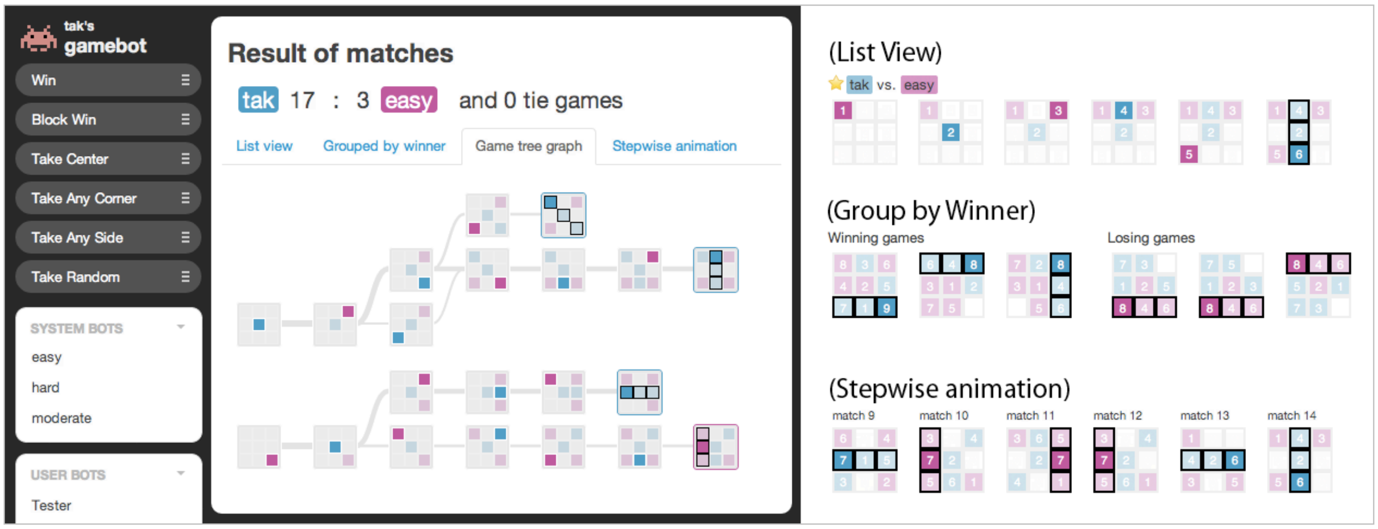


Figure 4. Review mode. By selecting an opponent (lower left) CTArcade runs 20 games and shows the scores and the patterns via four visualizations.

The Review section works as follows. First, by choosing an opponent in the list (lower left) it runs 20 matches and shows the results with four visualizations: 1) "List view" simply shows all the games in full detail; 2) "Group by winner" is useful for focusing on winning/losing games; 3) "Stepwise animation" is suitable to see temporal trends of all the games; and 4) "Game tree graph" provides an efficient way to find game play patterns without looking at hundreds of boards. If a learner found an interesting pattern using any of these visualizations, he/she can fix it by editing the current rules (upper left) which will rerun the matches immediately.

The visualizations are designed to be fairly easy to understand. However, the Game tree graph is a particularly interesting feature. The visualization first automatically creates groups of boards, which are identical after rotation and flip transformations. Second, edges connect pairs of groups if they happened consecutively in a game. Edge thickness represents the frequency of the transition. Finally, using Sugiyama's graph layout algorithm [22], it removes all the edge-crossings and aligns boards in a graphically pleasing way.

EVALUATION

We conducted a study to explore how playing Tic-Tac-Toe in the CTArcade environment might aid children in articulating CT skills compared to the typical method of playing via paper and pencil. Eighteen children were recruited to participate in the study that represented a diverse group in terms of: gender, age, ethnicity, and level of everyday interest in gaming. The participant group was 53% female and ranged between the ages of 10 and 15, with the average participant age being 11 years old. 58% of participants identified as Caucasian, 16% identified as Asian, and another 16% identified as being Multiracial. The participants all knew how to play the Tic-Tac-Toe game. The parents of the participant population responded to electronic ads placed in the Maryland and Massachusetts areas of the U.S. and were selected on a first-come-first-serve basis.

Each child met individually with a member of the research team and played Tic-Tac-Toe under two conditions: on paper (PAP) and in CTArcade (COMP). We used a within-subjects experimental design and the conditions were counterbalanced so that each participant was randomly assigned to experience the game conditions in a different order. For example, a child randomly assigned to the order PAP-COMP played the game on paper then in CTArcade (and vice versa for COMP-PAP).

Procedure

Sessions lasted approximately one hour. We first interviewed each child and collected background demographic information and data about their interests in gaming. Then each child played Tic-Tac-Toe, and we asked the participants to "think aloud" and explain their thought process during each move in the game. We were particularly interested in seeing how young children *articulate* CT skills under the various game play conditions, so the use of the "think aloud" protocol was salient in this context. When playing on paper, children played against a member of the research team (i.e. the interviewer). In CTArcade, a research team member explained how to use the game and children continued to articulate their thought process as they interacted with the environment. All sessions were audio recorded and transcribed.

CT Skill	Definition & Example Text
Algorithmic Thinking	Logical steps required to construct a solution to a given problem. *Most evident when participants discuss rules, thought processes, or priorities related to strategies.* e.g., *"No, first take any side and then take a corner." [referencing system rule names]*
Decomposition	Process of breaking down a large problem into smaller sub-problems or details. Explanation of an action in detail. e.g., *"I'm going to go in the bottom middle because it blocks you and makes my chance better of winning.[SIC]"*
Pattern Recognition	Relates board states, required actions, and events to other similar phenomena. *Participants can look at a board state and easily predict correct moves or outcomes.* e.g., *"You were probably about to pull something there, and then there, and obviously win the game." [identifying a board state]*
Pattern Generalization & Abstraction	Solving of problems of a similar type because of past experience solving this type of problem. *Participants are able to extract information from and discuss strategies.* e.g., *"It's sort of like that one, the first one where we just kept blocking each other's moves." [explaining multiple tie games]*
Unarticulated Instances	Performing an action within the game as a result of a thought process that could not be articulated. e.g., *"I'm going to take the bottom middle, to change it up." [referencing starting strategy]*

Table 1. Computational Thinking Definitions and Examples

In preparation for quantitative analysis, two of the paper authors first developed a codebook of reliable codes using an iterative process. In the first phase, four types of computational thinking skills employed in the design process were defined as they were expected to be found in the children's game play: Problem Decomposition, Pattern Recognition, Pattern Generalization, and Algorithmic Thinking. Further explanations of these skills are provided in Table 1. These skills were expected due to our previous experience observing children employing CT skills during game play [13].

In the second phase, both coders analyzed two of the interview transcripts of a participant playing on paper and in CTArcade. We compared coding notes and resolved any discrepancies in how codes were applied and interpreted. We found that one additional code was necessary during this initial review of the data. We found that in many cases, participants made moves in a game that clearly came from a thought process, but they could not articulate that thought process. Since Tic-Tac-Toe is a widely popular and common game, many moves are intuitive for children. While computational thinking is embedded in the children's thought process, they may not be able to articulate them, which is one goal of the CTArcade interface. Thus, we coded examples of children's gameplay where they did not articulate any thought process, but invested time in thought, as Unarticulated Instances.

Using this defined codebook, both coders analyzed an additional set of transcripts to determine the reliability of our definitions and interpretations. We continually iterated through coding and discussion until we achieved an acceptable level of reliability in the coding scheme (Krippendorf's Alpha = 0.71). After achieving a consistent level of understanding and application of the codebook, one of the authors coded the remaining transcripts.

We were also interested in a qualitative understanding of why the children enjoyed (or did not enjoy) CTArcade and their modes of interaction with the platform. The qualitative analysis followed the same iterative process used in developing reliable codes for the quantitative analysis. In the first phase, three types of explanations for Activity Interest were defined: Teaching Interest, Novel Gameplay, and Direct Control. Further explanation of these codes is provided in Table 2. Two of the authors reviewed four transcripts and coded the interviews using the appropriate tags. Upon completion, both coders compared coding notes, resolved any discrepancies in how codes were applied and analyzed the results. New codes were added as a result of this phase. After completion of the second phase both coders coded the remaining transcripts, compared notes, and resolved discrepancies.

Research Questions

Our study aims to consider several hypotheses and research questions. Our first aim is to examine whether playing the CTArcade version of Tic-Tac-Toe resulted in children better articulating aspects of CT skills employed in the scaffolded model design. The within-subjects, experimental design tested the following hypotheses:

Activity Interest	Definition (Type) & Example Text
Teaching Interest	Participant enjoyed the process of teaching, training, or mentoring their bot. **(COMP)** e.g., *"I think the computer, because you actually get to teach it how to play."*
Novel Gameplay	CTArcade is fun because it is a new and/or novel method of playing Tic-Tac-Toe. **(COMP)** e.g., *"Well, the computer was new, so it was interesting, because I never played like that before."*
Direct Control	Player's preference was to be able to have direct control over game actions **(PAP)** e.g., *"I actually got to play, and then in the computer the robot played it all for you."*
Technical Features	The features provided by the application (i.e. the storage and retrieval of moves, the choosing of custom character avatars, information visualization, etc.) e.g., *"I did not forget my moves." [referring to game history mode in the Trainer component]*
Thinking Opportunities	Provided opportunities to consider different methods of play, strategies, and/or thought processes **(COMP)** e.g., *"I like how you have to try and figure out what you can do to change it up." [strategy]*
Interactivity	The interactive nature of the application, either participant and bot and/or bot and bot **(COMP)** e.g., *"You get to play against other people or other people's robots."*

Table 2. Activity Interest Definitions and Examples

H1. Children will articulate more instances of CT skills (a. Algorithmic Thinking; b. Pattern Generalization & Abstraction; c. Problem Decomposition; d. Pattern Recognition) while using CTArcade compared to playing the original Tic-Tac-Toe.

H2. Children will describe more Unarticulated Instances while playing the original Tic-Tac-Toe compared to CTArcade Tic-Tac-Toe.

Since we used interviews and a think-aloud protocol to elicit the natural responses of children during game play, we also drew upon the rich qualitative data to understand two exploratory research questions:

R1 - Did children perceive CTArcade as being an enjoyable activity compared to paper?

R2 - What explanations were provided for the participants' preferences?

STUDY FINDINGS

H1., H2. Did Playing CTArcade help Children Articulate their Underlying Computational Thinking?

Result. We first examined the patterns of computational thinking skills that children articulated in both the paper Tic-Tac-Toe and CTArcade activities. We used repeated measures, one-way ANOVA to test our hypotheses. We also ran post-hoc analyses to determine if there were any ordering effects (e.g. going second always improved one's CT score), and found that ordering was not statistically significant. Figure 5 shows the average number of instances of specific CT skills articulated by the participants across activity types, which is the basis of this analysis.

On average, children articulated 5 instances of Algorithmic Thinking while playing CTArcade (M = 5.00, SD = 3.31). When playing on paper, children hardly ever articulated algorithmic thinking patterns (M = 0.22, SD = 0.43) although they may have been present in their unarticulated thinking instances. The data suggests that when children played in the CTArcade environment, they articulated substantially more instances of algorithmic thinking compared to playing on paper (H1.a supported). This difference was found to be statistically significant ($F_{1,17}$ = 33.4, $p < 0.001$). Unarticulated instances were most heavily found when children played on paper (M = 3.78, SD = 3.34) than when they played on the computer CTArcade (M = 0.39, SD = 0.50). This difference was also found to be statistically significant ($F_{1,17}$ = 22.49, $p < 0.001$) (H2 supported). We believe that our careful design of CTArcade helps young children transform their unarticulated instances into articulated Algorithmic Thinking.

An interesting finding was that children using CTArcade articulated significantly fewer examples (M = 3.61, SD = 1.94) of Pattern Recognition compared to playing on paper (M = 6.72, SD = 2.11) ($F_{1,17}$ = 61.99, $p < 0.001$). There were no statistically significant differences in Pattern Generalization or Problem Decomposition between playing on paper or in CTArcade. These findings make sense as the children participants spent the majority of their time using CTArcade in the Trainer mode, which explicitly scaffolds them to teach their virtual characters to play via sequences of game rules (algorithmic thinking patterns). Therefore, it is not surprising that there were more instances of algorithmic thinking in the CTArcade interface.

Discussion. For the finding related to pattern recognition, we posit that this CT skill was observed in the paper version more often because of the nature of the conditions. Pattern recognition is readily accessible when playing on paper. For example, when playing Tic-Tac-Toe on paper, most child participants were able to easily justify their move by recognizing the pattern on the board. For example,

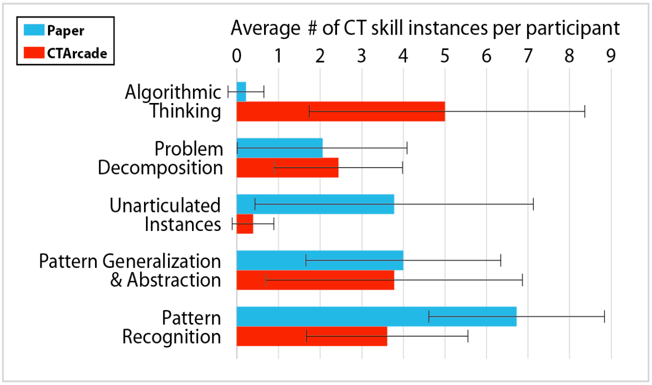


Figure 5. Average numbers of articulated CT skill instances by activity type

a 10 year old male participant attributed his moves to the recognition of several patterns saying, "to block you [the interviewer] and since I have no other way to win, I'll use the top center to block and now there's no way for us to win." However, in the Trainer mode, the participant was teaching a virtual character, which predominantly scaffolded Algorithmic Thinking over Pattern Recognition.

In Trainer, a child would play one step, and the bot typically articulated the pattern recognition when it asked the user why they made a certain move (removing the need for the child to articulate it to us in the interview). Another possible explanation for this finding is that we observed a lot of overhead associated with learning how to use the computer application versus the intuitiveness of playing on paper. In fact, we introduce an entirely new playing paradigm in terms of Tic-Tac-Toe, and there may have been less opportunity to observe Pattern Recognition as the children were busy learning how to use CTArcade for much of the sessions. Overall, our findings suggest that the carefully scaffolded interface in CTArcade influenced the children to focus more on Algorithmic Thinking, which almost never happened in the paper game.

R1. Did children perceive CTArcade as being an enjoyable activity compared to paper?

Result. Our qualitative analysis focuses on assessing the enjoyment experienced by the children as it related to using CTArcade. This qualitative understanding is important because it demonstrates whether or not we have successfully created an enjoyable gaming experience as opposed to an experience that feels more like an educational activity. 16 out of 18 (89.0%) participants reported that they enjoyed CTArcade. 1 out of 18 (5.5%) participants reported that they did not enjoy CTArcade. Finally, 1 out of 18 (5.5%) participants reported that they were neutral on whether or not they enjoyed CTArcade. These findings suggest that CTArcade is to some degree an enjoyable activity.

Once we established whether or not using CTArcade was an enjoyable activity, we then asked participants to compare CTArcade to the paper activity. We expected that a minority of participants who thought CTArcade was fun would still prefer the paper activity. We also expected that participants who did not enjoy CTArcade would report preferring paper as well. 13 out of 18 (72.3%) participants preferred CTArcade to paper, and 2 (11.1%) enjoyed both activities equally. Only 3 out of 18 (16.6%) participants preferred paper to CTArcade.

Discussion. Overall we view these findings as favorable support for further development of CTArcade. We posit that we may be able to facilitate more enjoyable game play and CT learning by increasing the library of games available through the platform. By including more games we would be able to provide different game types (i.e. strategy, puzzle, etc.), varying levels of cognitive challenges, and others motivations for playing (e.g. competitive and collaborative games) to capture the interest of more users.

R2. What explanations were provided for the participants' preferences?

Result. When we asked the children to explain their preferences, most participants gave two primary reasons for their experiences. A majority of our participants reported that the primary factor for enjoying CTArcade was the teaching aspect of the platform. For example, a 12-year old male participant stated that he enjoyed CTArcade "because you actually get to teach it how to play." This finding relates to prior research which finds that teaching others can be an effective mechanism for improving the skills of the learner [3]. We reason that the teaching, or training, aspect of CTArcade has helped generate the positive experiences of our child participants.

In addition to teaching, participants provided a variety of other explanations for their enjoyment. 6 out of 18 children commented that the novelty, or the new way, of playing the game was appealing. For example, a 12 year old male participant stated that,

> *"…it's [CTArcade] generally more fun than just playing the same old game [Paper] with someone I know because there's nothing special to it."*

It is important to note that not every participant enjoyed CTArcade. Of the few children who reported not enjoying CTArcade, the main reason was related to their desire for direct control in the outcome of the games being played. Rather than training a bot to play, these children had a deeper desire to affect the outcome of the game through their own direct actions. For example, a 10 year old male participant explained,

> *"you get [it] to do what you want. You don't have to...When you have a player on the computer it just takes its own moves. Of course I taught it the moves, but it is better when you yourself can go where you want to [SIC]."*

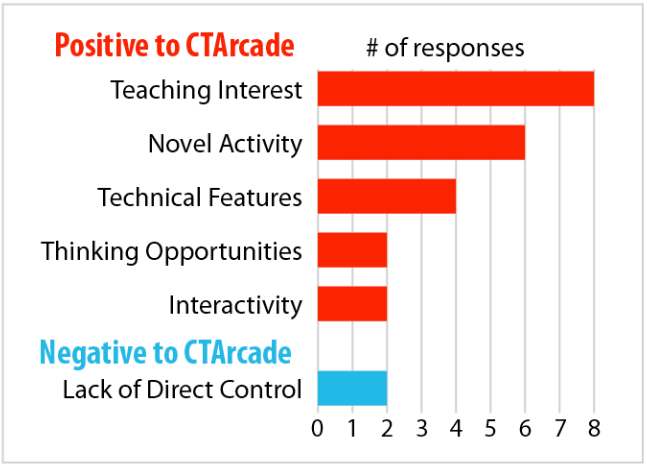


Figure 6. Explanations of Preferences

Discussion. The rationales behind the experience are interesting to us for several reasons. The participants who preferred the paper activity suggest that our framing of CT education makes some users feel that they lack control, which leads to the paper experience being superior. A number of participants preferred the computer activity because it is a new and novel experience, which highlights that further study is required to determine that CTArcade remains interesting after prolonged exposure. Finally, we found it encouraging that most children predominantly enjoyed the teaching aspect of the platform.

LIMITATIONS

One limitation of the study is that the half an hour was insufficient for the participants to fully experience the scaffolded model. The children spent most of their time with the first layer of the scaffolds (list of rules) in the Trainer mode. Moreover, the list of rules is present across the other modes of Tic-Tac-Toe, helping to direct learners back to algorithmic thinking whenever they want to edit their strategies. These situations helped Algorithmic Thinking dominate the study results and do not indicate that CTArcade cannot foster improvements in other CT skills. Perhaps more time spent with other features in CTArcade, that were explicitly designed for other CT skills, would also result in more articulation of those particular thinking patterns.

Furthermore, a single game is not generalizable enough to make conclusions about our approach to CT education. Tic-Tac-Toe is a relatively simple game and near-perfect strategies were often quickly found. The model was mainly focusing on algorithmic thinking and was easily scaffolded. These attributes of Tic-Tac-Toe made it an ideal candidate for this exploratory study and the results are positive; however, further work is required.

FUTURE WORK

More Games and Models

To draw more general conclusions, we will develop and test more games. The good news is that the CTArcade platform can be made more applicable to a wide variety of games with the development of a few more scaffolding models. The current rule-based model is widely applicable to simple strategy games such as Connect Four or Mancala. To support arcade/sport games (i.e. Space Invader, Robot soccer) we can build a new event-driven strategy model. An additional parametric configuration model can cover many simulation games (e.g. Game of Life, Prisoner's dilemma).

Social Engagement

We found that the framing of teaching virtual characters provided an excellent motivation for articulating one's tacit knowledge. Other social activity (e.g. such as two learners playing side-by-side on adjacent computers, playing together to build a single solution, or building more collaborative games) will likely provide further, and alternative, motivations to learn.

Integrated Learning Environment

Although CTArcade is originally designed as a self-contained gaming environment, formal educational activities, such as in-classroom lecture or massive open online courses (MOOC) [14] can provide well-grounded knowledge about CT. To make this integration powerful, each game should have a specific topic (i.e. recursion, regular-expression, or prisoner's dilemma) rather than fostering common CT skills. Designers of integrated learning environments should also consider how to encourage Student-Teacher interaction as well.

Formal evaluation and longitudinal study

Future formal evaluations will need to be structured in a manner that allows enough time for participants to access all of the components of CTArcade in a meaningful way without a lot of guidance from the interviewer. Additionally, a larger representative sample of participants is necessary to be able to analyze the results in terms of differences between gender, ethnicity, and age before we can conclusively say that these results will scale to the general population.

DESIGN IMPLICATION

The approach and design of the CTArcade platform was originally conceived as a way to build a library of games that teach CT. It is important to note that this approach may also be helpful when attempting to teach other non-CT concepts. We propose several design implications for educational games and environments.

Focus on the topic that is best-supported by the strategy template and the most interactive component. While our Tic-Tac-Toe game of CTArcade aims to support multiple CT skills, we found that algorithmic thinking dominated the results and the learner's experience. We believe this occurred because algorithmic thinking is directly supported by both the rule-based list model and the UI. We learned a lesson here; designing a game to equally engage multiple skills could be more challenging than creating multiple games with distinct learning objectives.

Use Teaching/Helping others as a motivator. Many of our participants reported that they enjoyed the teaching aspect of training their virtual characters. As a result, we believe that teaching is a positive attribute of CTArcade. Additionally, teaching provides excellent motivation for internalizing and improving skills [3].

Determine the best method of providing help to the learner and implement it. CTArcade was conceived as a platform to teach children CT skills in a classroom setting and evolved into being accessible to the general public through an online web application. However, we consistently noticed during various pilot tests and this user study that we had to provide instruction to participants on general operations and conceptual elements. The designers of systems like CTArcade have several choices in terms of how to provide help. One method is through careful construction of the UI allowing for the utilization of tool tips and the highlighting of critical information. Other methods might be providing written help guides or providing a detailed tutorial feature that slowly introduces learners to the various parts of the application while they play and use information hiding strategies (i.e. hiding pieces of the application that have not been explained yet). Designers could also take a hybrid approach by using combinations of these methods. Finally, we do not want to discount the potential value of having an instructor present while learners use the system.

CONCLUSION

As the first generalizable platform that bridges the gap between natural game play and educational programming, this paper described CTArcade, a gaming environment that enables children to articulate their innate CT skills and conceptualize them as formal logic. Tic-Tac-Toe, the first game developed in the CTArcade platform, features a novel scaffolding for learning. From the exploratory user study we found that CTArcade helped the children articulate more algorithmic thinking skills compared to playing naturally on paper. Based on our experiences developing CTArcade, we provide some guidelines for making educational games and educational gaming environments. We believe that the CTArcade platform and our findings are relevant not only to HCI researcher interested in CT education but also to professional designers working in end-user programming and user-experience.

REFERENCES

1. Basawapatna, A., Koh, K. H., Repenning, A., Webb, D. C., & Marshall, K. S. (2011). Recognizing computational thinking patterns. In Proceedings of the 42nd ACM Technical Symposium on Computer Science Education. New York: ACM.

2. Berland, M., and Lee, V. R., (2011). Collaborative strategic board games as a site for distributed computational thinking. International Journal of Game-Based Learning 1, 2 (2011), 65-81.
3. Chase, C., Chin, D. B., Oppezzo, M., & Schwartz, D. L. (2009). Teachable agents and the protégé effect: Increasing the effort towards learning. Journal of Science Education and Technology.
4. Cypher, A., Halbert, D. C., Kurlander, D., Lieberman, H., Maulsby, D., Myers, B. A., & Turransky, A., (Eds.). (1993). Watch what I do: Programming by Demonstration. MIT Press, Cambridge, MA, USA.
5. Frager, S., Stern, C. (1970). Learning by Teaching. The Reading Teach, 23(5), 403 - 405.
6. Google. (n.d.). Exploring computational thinking. Retrieved on April 3, 2011 from http://www.google.com/edu/computational-thinking/.
7. Holbert, N. R., and Wilensky, U., (2011). Racing games for exploring kinematics: A computational thinking approach. Paper presented at AERA 2011, New Orleans, LA, USA
8. Kato, H., & Ide, A. (1995). Using a game for social setting in a learning environment: AlgoArena; a tool for learning software design. In The first international conference on Computer support for collaborative learning (CSCL '95), NJ, USA, 195-199
9. Kelleher, C. & Pausch, R., (2005). Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers. ACM Comput. Surv. 37, 2 (June 2005), 83-137
10. Kelleher, C., Pausch, R., & Kiesler, S. (2007). Storytelling alice motivates middle school girls to learn computer programming. In CHI '07: Proceedings of the SIGCHI Conference on Human factors in Computing Systems. New York: ACM.
11. Kodu. (n.d.). Retrieved on April 18, 2011 from http://research.microsoft.com/en-us/projects/kodu/
12. Lee, I., Martin, F., Denner, J., Coulter, B., & Allan, W., et al. (2011). Computational thinking for youth in practice. ACM Inroads, 2(1), 32-37.
13. Lee, T. Y., Mauriello, M. L., Ingraham, J., Sopan, A., Ahn, J., and Bederson, B. B. (2012). CTArcade: learning computational thinking while training virtual characters through game play. In Proceedings of the 2012 ACM annual conference extended abstracts on Human Factors in Computing Systems Extended Abstracts (CHI EA '12). ACM, New York, NY, USA, 2309-2314
14. McAuley, A., Stewart, B., Siemens, G. & Cormier, D. (2010). The MOOC Model for Digital Practice. Retrieved from http://www.elearnspace.org/Articles/MOOC_Final.pdf. (20th May 2011).
15. Rinard, M. C., (2012). Example-driven program synthesis for end-user programming: technical perspective. Commun. ACM 55, 8 (August 2012), 96-96. DOI=10.1145/2240236.2240259 http://doi.acm.org/10.1145/2240236.2240259
16. Myers, B. A., Pane, J. F., & Ko, A.. 2004. Natural programming languages and environments. Commun. ACM 47, 9 (September 2004), 47-52.
17. O'Kelly, J., and Gibson, J. P., (2006). RoboCode & problem-based learning. ACM SIGCSE Bulletin, 38(3):217, June 2006.
18. Pane, J.F., Myers, B.A., & Miller, L.B., (2002). Using HCI Techniques to Design a More Usable Programming System, 2002 IEEE Symposia on Human Centric Computing Languages and Environments (HCC'02). Arlington, VA, September 3-6, 2002. pp. 198-206
19. Papert, S. (1971). Teaching children thinking. Retrieved on April 1, 2011 from http://dspace.mit.edu/handle/1721.1/5835.
20. Papert, S. (1993). Mindstorms: Children, computers, and powerful ideas. Basic Books, New York, NY, USA
21. Resnick, M. Maloney, J., Monroy-Hernandez, A., Rusk, N., & Eastmond, E. et al. (2009). Scratch: Programming for all. Communications of the ACM, 52(11), 60-67.
22. Sugiyama, K., Tagawa, S., and Toda, M. (1981). Methods for visual understanding of hierarchical system structures. IEEE Transactions on Systems, Man, and Cybernetics, 11(2):109–125, February 1981.
23. Wing, J. M. (2006). Computational thinking. Communications of the ACM, 49(3), 33-35.