International Journal of Child-Computer Interaction 2 (2014) 26–33

Contents lists available at ScienceDirect

International Journal of Child-Computer Interaction

journal homepage: www.elsevier.com/locate/ijcci

CTArcade: Computational thinking with games in school age children

Tak Yeon Lee [a,b,*], Matthew Louis Mauriello [a,b], June Ahn [a,c,d], Benjamin B. Bederson [a,b,c]

[a] Human-Computer Interaction Lab, University of Maryland, College Park, MD 20742, USA
[b] Computer Science Department, University of Maryland, College Park, MD 20742, USA
[c] iSchool, University of Maryland, College Park, MD 20742, USA
[d] College of Education, University of Maryland, College Park, MD 20742, USA

ARTICLE INFO

Article history:
Received 22 July 2013
Received in revised form
14 June 2014
Accepted 28 June 2014

Keywords:
Computational thinking
Learning
Games
Human factors
Design

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 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.

1. Introduction

Computational thinking (CT) is a set of thinking patterns that includes understanding problems with appropriate representation, reasoning at multiple levels of abstraction, and developing automated solutions [1]. CT has become a fundamental skill and should be cultivated by everyone, not just computer scientists [2,3,1]. Researchers in other disciplines discover breakthroughs by using the same CT skills valued by computer scientists. Many everyday activities using information technology can be done more efficiently when the person performing them has CT skills. However, one challenge in fostering CT is the lack of opportunities to improve one's CT skills.

Building one's own game has been a popular task in educational programming environments. Papert [4] coined the term "constructionism" and observed that learning is most effective when learners construct a meaningful product. Researchers have contributed foundational work to create visual programming environments that allow children to create their own games by manipulating graphical (and tangible) blocks rather than relying on textual syntax. For instance, AgentSheets [5] helped students recognize patterns of CT while designing games, and allowed them to apply the patterns in science simulations. Storytelling Alice [6] taught basic programming constructs to middle school girls by giving them tools for creating interactive animations. HANDS [7] showed that the usability of programming environments is crucial for children to learn CT skills. Although game creation is an effective way to learn basic CT skills, building a playable game from scratch is still challenging for novice programmers. This level of challenge means that only strongly-motivated children can build playable games that involve sophisticated CT skills even though these skills are readily available in the natural game play exhibited by their peers.

Learning scientists and educational experts have acknowledged that children playing various genres of games have shown a variety of sophisticated CT patterns that emerge in their game play. For example, racing games encourage children to internalize one-to-one mapping of game action to kinematic concepts [8,9]. Different game activities are associated with various characteristics of CT skills [10]. However, a formative study [11] found out that simply playing games does not help children articulate CT skills unless the game is equipped with carefully designed support for articulating CT skills.

* Corresponding author at: Human-Computer Interaction Lab, University of Maryland, College Park, MD 20742, USA. Tel.: +1 202 330 2506.
E-mail addresses: tylee@umd.edu (T.Y. Lee), mattm@cs.umd.edu (M.L. Mauriello), juneahn@umd.edu (J. Ahn), bederson@cs.umd.edu (B.B. Bederson).

http://dx.doi.org/10.1016/j.ijcci.2014.06.003
2212-8689/

Another approach is to combine the advantages of natural game play and educational programming environments by using a programmable game character strategy as done by a small number of simulation game platforms. One example of this approach is AlgoArena [12], which enabled students to program their own wrestler characters in order to win matches against other wrestlers. RoboCode [13] is another game environment where each player develops AI using the Java language to manoeuver a robot tank in order to search out and destroy other tanks. This problem-driven learning has several advantages. First, it allows students to focus on advanced algorithmic logic without requiring them to build basic functionality of the games. Second, when the children develop the AI, they transform tacit knowledge about game play into formal logic, which is a common way of learning CT skills. Third, social interaction on the tournament servers provides extra motivation. Unfortunately, currently available educational games, requiring professional programming skills, are not accessible to novice programmers.

In this paper, we describe CTArcade—an educational game environment that enables children to articulate CT-related thinking patterns while playing games. We developed Tic-Tac-Toe as the first game to utilize the CTArcade environment. We report a "think aloud" user study that observes 18 children playing the CTArcade version of Tic-Tac-Toe and the original version on paper. We compared the gameplay of these environments through (1) quantitative analysis of code counts for instances of different kinds of CT skills, and (2) descriptive examples of the CT skills utilized by the children. 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 an exploratory user study with 18 children that shows evidence that scaffolded gameplay in CTArcade helped them articulate more algorithmic thinking skills compared to playing naturally on paper.

2. Design process

The CTArcade design was strongly motivated by a formative study [11] that examined how young children (aged 7–11) utilized computational thinking skills as they play the paper-based game, Connect Four. We observed that children naturally used complex CT skills (e.g. recognizing the winning and losing conditions of the board). However, they expressed difficulty developing algorithmic representation of their gameplay strategy. The study provided design considerations about how to use games to teach CT and we directly applied them in the design of CTArcade. We then designed the initial set of Tic-Tac-Toe strategy and AI agents based on the strategy created during the formative study. After building the first working prototype, we iteratively conducted pilot tests and redesigned it with children. We had three goals in the design of CTArcade:

- *Integrate tacit knowledge within the game interface*. Decomposing innate thinking into abstract representations is difficult for children. Providing a natural way of formalizing tacit knowledge during game play has a critical place to help learners explicitly link their game actions to CT. The CTArcade Trainer employs a mixed-initiative approach [14] that infers and suggests potential strategic rules corresponding to the most recent move made by the learner. Using this method allows the learner to either pick the right rules or directly specify a new rule in the creator mode.
- *Moving from concrete to abstract computational thinking*. In order to help learners transform their tacit knowledge about game play into formal logic, CTArcade guides them to create rules for specific game situations first and then to generalize them later. Such progress requires the process of *concreteness fading* [15], CTArcade provides features that gradually reduce the salience of the specific game situation to an abstract strategy.
- *Reduce cognitive load and split attention*. Having separate elements of information (e.g. a game board and a rule sheet) to pay attention to would significantly raise the cognitive workload for learners. We designed the CTArcade platform to minimize the split attention effect.

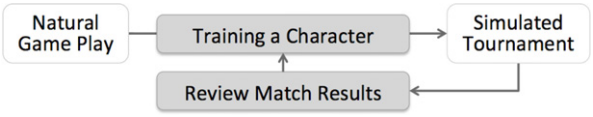

Fig. 1. CTArcade framework focuses on creating a learning loop for articulating CT skills, where children can train their own virtual characters and review tournament results.

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

- *Templates* for game strategy help learners 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.
- *Tournament feedback* helps learners to evaluate their own strategies and learn from other people, which can lead them to iterative reflections and fine-tuning of one's gameplay logic.
- *Visual analytic tools* enable learners to find patterns from a number of tournament results.

3. CTArcade: designing to teach computational thinking

CTArcade is a web-based educational gaming environment that extends simple games with scaffolded learning activities (see Fig. 1). In CTArcade children start by naturally playing Tic-Tac-Toe against their own game characters, which know very little about game play strategies. While playing the game, a child will conceptualize his/her game play into abstract algorithmic rules that the character will then follow. The programmed characters will then join the tournament server and play against other characters. The children can review results, recognize patterns of losing matches with four different visualizations, and make refinements to their strategy in iterative manner. The learning loop highlights 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. However, these skills are applied rapidly in a natural context and children do not inherently conceptualize their play as abstract logic. On the other hand, visual programming environments help to lower the technical barrier of programming, but do not necessarily take advantage of natural activities requiring CT skills. Our goal with CTArcade is to complement these approaches by helping young children internalize CT skills while converting their natural game play into a formal strategy via a scaffolded model. The following sections illustrate components of CTArcade learning loop in detail.

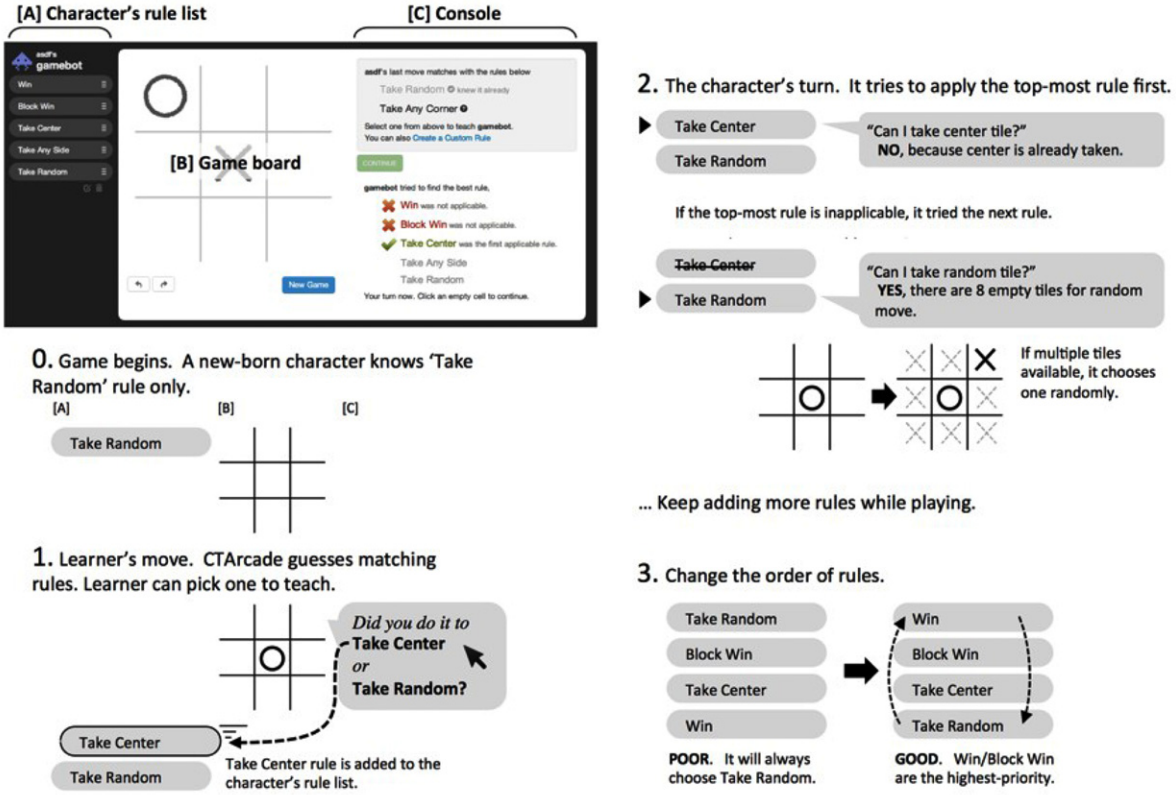

Fig. 2. Trainer UI consists of: (A) list of ordered rules that the character will apply for each move; (B) Tic-Tac-Toe game board; and (C) console showing all the messages that include suggestions of predefined rules matching with the learner's latest move.

3.1. Training a character

The learning loop starts from the Trainer, where children train their own characters while playing Tic-Tac-Toe. Fig. 2 illustrates the step-by-step procedures of the training process. For each move made by learners, CTArcade suggests matching predefined standard rules,[1] which were based on our formative user study. The learner can pick one of the suggested rules, and add it to the list of rules that the character will apply. In doing so, they train a virtual character which they can eventually play against themselves or enter into a virtual tournament. By selecting the rule closest to their own thinking, the learner is encouraged to reflect upon why they moved the way they did—which brings out their otherwise tacit algorithmic thinking.

While the above procedure is convenient to train the character with predefined rules, often learners might have custom rules in their minds. With the custom rule creator shown in Fig. 3, learners can create algorithmic rules from board states using concreteness fading [15]. First, when the learner opens the creator, the current game board is shown. The learner can simplify the board by putting *question marks* in the squares that are irrelevant to the rule he/she wants to create. A double-circle indicates the rule's outcome (i.e., which cell is filled in). After defining an exemplary board, the learner can choose *transform operations* (Row/Column Permutation, Flipping, and Rotation) in order to apply the rule in related situations. Using this two-step method, we tried to engage learners to develop their CT skills including pattern recognition and abstraction/pattern generalization.

[1] 6 standard algorithms (*Take Random, Take Center, Take Any Corner, Take Any Side, Block Win*, and *Win*) are provided.

3.2. Simulated tournament

Debugging and performance testing are essential activities in CT learning. In CTArcade, trained characters join in a tournament against other characters, which was inspired by tournament servers in popular examples such as Robocode [13] and AlgoArena [12]. In order to bootstrap the tournament before other user-trained characters joined, we created *easy*, *medium*, and *hard* characters based on the experience of the formative study. The *easy* character randomly takes any tile unless it can win with a single move. The *moderate* character is a little smarter than the *easy* character; the *moderate* character will try to block the opponent's winning move. The *hard* character uses all the predefined rules in an optimized order, which is very hard to win against. Competitive learning environments allow learners to demonstrate their programming abilities and have fun [13]. CTArcade lowers the barrier to engaging in match play, such that even young children, without any programming knowledge, can participate.

3.3. Reflect on match result

Reflecting on 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 CT skills. CTArcade provides four visualizations (as illustrated in Fig. 4) that enable learners to recognize patterns from at least 20 winning and losing games. By making opportunities for identifying patterns easier, the learner is able to make generalizations about Tic-Tac-Toe strategy.

The review process works as follows. First, a learner chooses any opponent in the list (lower left) to run 20 individual matches. The system then shows the match results available in four visualizations: (1) "List view" simply lists all the match results. While the

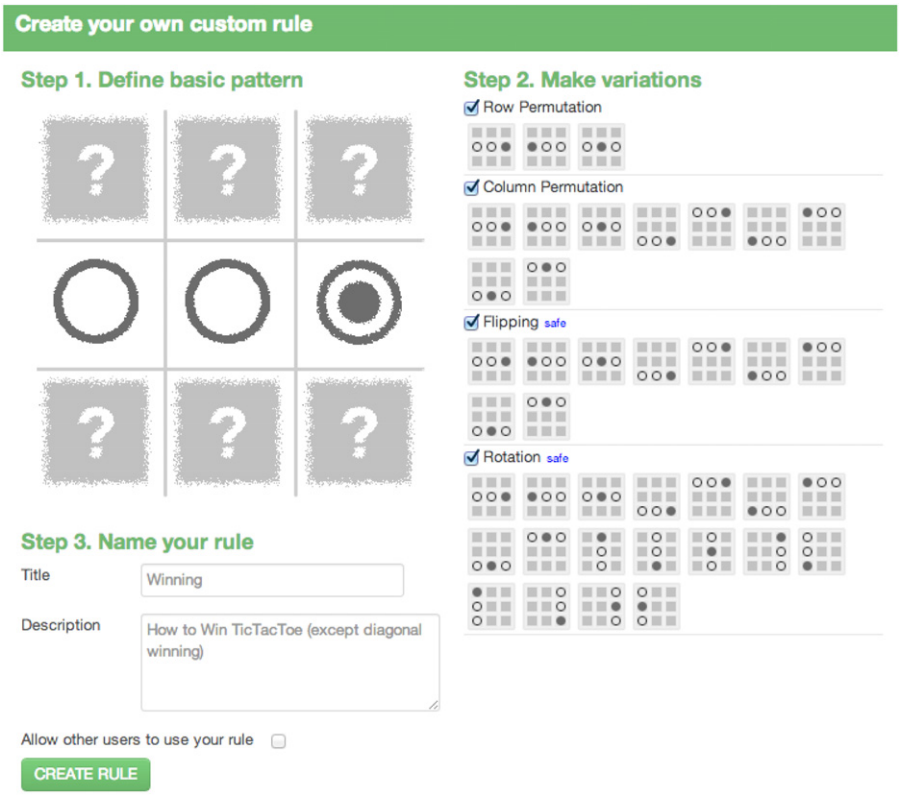

Fig. 3. Custom rule creator allows learner to create new rules by: Step (1) defining a basic pattern; and Step (2) applying the pattern across multiple variations.

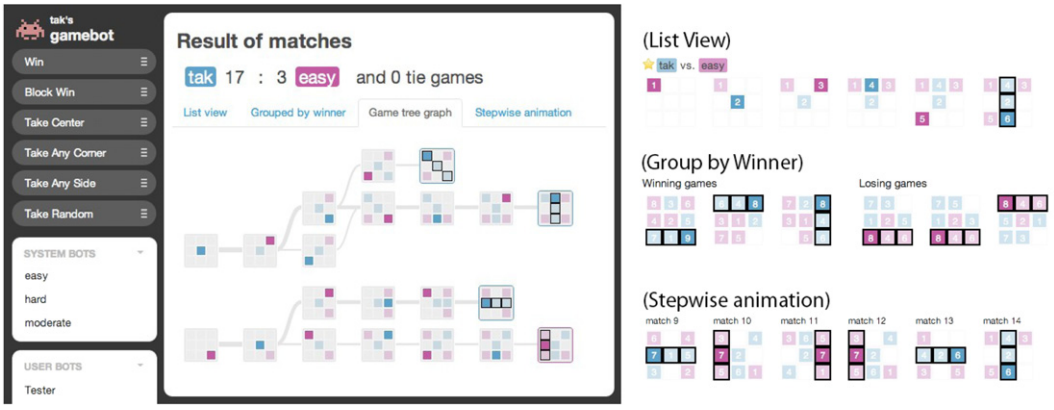

Fig. 4. Review mode. By selecting an opponent (lower left) CTArcade runs 20 games and shows the scores and the patterns via four visualizations.

view is easy to understand an individual match, telling how the character won or lost in general is very challenging; (2) "Group by winner" shows only the final states of the games grouped by winner; (3) "Stepwise animation" provides two (previous/next) buttons for navigating the previous/next moves, which are helpful to see temporal trends of all the games; and finally (4) "Game tree graph" automatically creates groups of boards, which are identical after rotation and flip transformations. Then it draws a graph whose edges connect pairs of boards if they happened consecutively in a game. Thickness of each edge represents the frequency of the transition. Using Sugiyama's graph layout algorithm [16], it removes all the edge-crossings and aligns boards in a graphically pleasing way. 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.

4. Evaluation

4.1. Method

We conducted an explorative study to observe how playing Tic-Tac-Toe in the CTArcade environment might aid children in articulating CT skills compared to the typical method of playing Tic-Tac-Toe on 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 an average age of 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

Table 1
Comparing two versions of Tic-Tac-Toe on paper (PAP) and in CTArcade (COMP).

	On paper (PAP)	In CTArcade (COMP)
Activity	Typical Tic-Tac-Toe	Playing Tic-Tac-Toe and training virtual characters
Whom they play with	Against human player	Against/With virtual characters
What they think aloud	'Why' the learner played the move	Explaining Tic-Tac-Toe rules to virtual characters

Table 2
Computational thinking definitions and examples.

CT Skill	Definition and example text
Algorithmic thinking	Logical steps required for constructing 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 then there, and obviously win the game." (Identifying a board state)*
Pattern generalization and abstraction	Solving of problems of a similar type because of the 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)*

Maryland and Massachusetts areas of the US and were selected on a first-come, first-serve basis.

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. 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) in Table 1. 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. 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.

4.2. Research questions

Our first aim is to compare the two versions of Tic-Tac-Toe in terms of CT learning effect. The within-subjects, experimental design tested the following hypothesis:

H1. *Children will articulate more instances of CT skills (a. Algorithmic Thinking; b. Pattern Generalization and Abstraction; c. Problem Decomposition; d. Pattern Recognition) while playing COMP compared to PAP.*
H2. *Children will describe more Unarticulated Instances of CT skills while playing PAP compared to COMP.*

Although the hypothesis implies COMP would have stronger learning impact, we rather expected both PAP and COMP versions to have their own strength and weakness. 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 (COMP) as being a more enjoyable activity compared to paper (PAP)*?
R2. *What explanations were provided for the participants' preferences*?

4.3. Analysis

In the first phase of the analysis, we used an inductive, grounded theory approach [17] to code the transcripts. Two of the paper authors developed a codebook of codes using an iterative process. 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 2. These skills were expected due to our previous experience observing children employing CT skills during game play [11].

After coding types of articulated CT skills, 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

Table 3
Activity interest definitions and examples.

Activity interest	Definition (Type) and 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."*

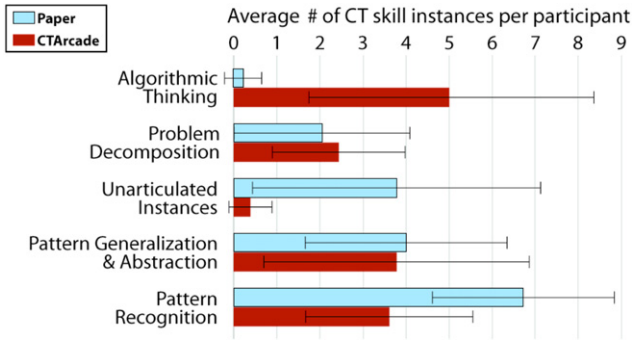

Fig. 5. Average numbers of articulated CT skill instances by activity type.

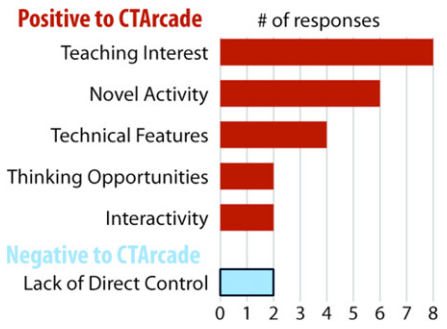

Fig. 6. Explanations of preferences.

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. Additional codes were added as a result of this iterative process. Further explanation of these codes is provided in Table 3.

5. Findings

5.1. Articulated instances of computational thinking

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. Fig. 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.

The children articulated an average of 5 instances of algorithmic thinking in COMP ($M = 5.00$, $SD = 3.31$). While in PAP, children hardly ever articulated algorithmic thinking patterns ($M = 0.22$, $SD = 0.43$). The data suggests that the CTArcade helped the children articulate substantially more instances of algorithmic thinking compared to typical Tic-Tac-Toe game (H1.a supported). The difference was found to be statistically significant ($F1, 17 = 33.4$, $p < 0.001$).

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 ($M = 6.72$, $SD = 2.11$) ($F1, 17 = 61.99$, $p < 0.001$). In the PAP setting, most participants naturally justified their moves by recognizing the pattern on the board. For instance, 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 COMP setting, a child would play one step, and the system then automatically completed the pattern recognition job and suggested the matching rules. This automation effectively removed the need for the children to verbalize recognized board patterns for every move, and instead focused on articulating algorithmic thinking. These findings make sense as the Trainer mode allows the learners to train their characters via sequences of game rules (algorithmic thinking patterns) instead of explaining recognized board patterns.

On the other hand, 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 ($F1, 17 = 22.49$, $p < 0.001$) (H2 supported). The findings suggest that the careful design of CTArcade helps young children transform their unarticulated instances into articulated Algorithmic Thinking.

Other types of CT skills (e.g. Abstraction, and Problem Decomposition) that are not explicitly supported in the Trainer mode did not show statistically significant differences between PAP and COMP settings. 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 most of the sessions. A further study involving the Tournament and the Review mode would shed deeper light on articulating other types of CT skills though.

5.2. Enjoyment of game play

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 playing CTArcade, 1 out of 18 (5.5%) was neutral, and only 1 (5.5%) participant reported that COMP was not enjoyable. It suggests 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 COMP and PAP settings. 13 out of 18 (72.3%) participants preferred COMP to PAP, and 2 (11.1%) enjoyed both equally. Only 3 out of 18 (16.6%) participants preferred PAP to COMP. 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.

As shown in Fig. 6, a majority of our participants reported that the primary factor in their enjoyment of 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 the prior study, which finds that teaching others can be an effective mechanism for improving the skills of the learner [18]. We reason that the teaching 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 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. 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)".

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 novel experience, which highlights that further study is required to determine whether CTArcade remains interesting after prolonged exposure. Finally, we were encouraged by the fact that most children enjoyed the teaching aspect of CTArcade.

6. Design suggestions

Our experience of developing and testing CTArcade Tic-Tac-Toe leads us to make the following suggestions for educational games and environments for computational thinking.

Focus on the educational goal that is best supported by the activity. While the original Tic-Tac-Toe game articulates mostly pattern recognition, we observed algorithmic thinking dominated the results and the learner's experience in CTArcade. We believe this occurred because the rule-based list model and the UI of CTArcade successfully reduced the need of pattern recognition and helped children focus on algorithmic thinking. As the human mind has limited cognitive capacity, an educational game environment should be designed to minimize split attention effect and focus on a specific CT skill.

Use teaching and 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 [18].

Determine the best method of providing help to the learner and implement it. We consistently noticed that we had to provide instruction to participants on general operations and conceptual elements in CTArcade. 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. Designers could also take a hybrid approach by using combinations of these methods.

7. Future work

Longitudinal study

First, the 1-hour session was not enough to experience all the components of CTArcade such as the tournament and the match review modes. In order to generalize the finding about algorithmic thinking to other CT skills, future evaluations need to be structured in a manner that allows enough time (at least 2-hours, preferably multiple sessions) for participants to get familiar with all of the components of CTArcade. In addition, we would develop our own evaluation method of CT skills, and use it to evaluate how much learners have improved CT skill level while playing it.

Comparative evaluation with traditional games

We compared CTArcade with the paper-based Tic-Tac-Toe game and found that each condition articulates different CT skills. In order to gain deeper insights about the effectiveness of the training mode, a future study would need to compare the two versions of CTArcade with and without the training mode. Other components such as the tournament and the review mode can also be evaluated by similarly setting up experimental conditions with and without these components.

More games and models

Tic-Tac-Toe was the first game we extended in CTArcade platform. In order to show generalizability of the CTArcade platform, we hope to develop more games with diverse interaction models. For example, 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) CTArcade needs to support event-driven strategy model. Game theory model can cover more simulation games such as Game of Life, and Prisoner's Dilemma. With these models, CTArcade can be made more generalizable to other games and learning objectives of increased complexity.

Social engagement

The current CTArcade platform provides a basic competitive environment using the tournament server and the leaderboard. To provide stronger motivation, the next step would be to try collaborative and synchronous social activities that require no additional functionality. For instance, if two or more children played the game together, they could get more instant feedback on their revised AIs. In synchronous settings, children can also discuss about their strategies and articulated CT skills.

Integrated learning environment

Although CTArcade was originally designed as a self-contained gaming environment, formal educational activities, such as in-classroom lecture or massive open online courses (MOOC) can provide well-grounded knowledge about CT [19]. 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.

8. Conclusion

As the first platform (to our knowledge) that bridges the gap between natural game play and computational thinking, this paper described CTArcade, a gaming environment that enables children to articulate their innate CT skills and conceptualize them into algorithmic rules. Tic-Tac-Toe, the first game developed in the CTArcade platform, features explicitly designed 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 in 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 researchers interested in CT education but also to researchers and designers working in areas of children, learning, and technology.

References

[1] J.M. Wing, Computational thinking, Commun. ACM 49 (3) (2006) 33–35. http://dx.doi.org/10.1145/1118178.1118215.
[2] A.A. DiSessa, Changing Minds: Computers, Learning, and Literacy, MIT Press, Cambridge, Mass., 2001.
[3] M. Guzdial, Education: paving the way for computational thinking, Commun. ACM 51 (8) (2008) 25–27. http://dx.doi.org/10.1145/1378704.1378713.
[4] S.A. Papert, Mindstorms: Children, Computers, and Powerful Ideas, second ed., Basic Books, 1993.
[5] A. Basawapatna, K.H. Koh, A. Repenning, D.C. Webb, K.S. Marshall, Recognizing computational thinking patterns, in: Proceedings of the 42nd ACM Technical Symposium on Computer Science Education, ACM, New York, NY, USA, 2011, pp. 245–250. http://dx.doi.org/10.1145/1953163.1953241.
[6] C. Kelleher, R. Pausch, S. Kiesler, Storytelling alice motivates middle school girls to learn computer programming, in: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, ACM, New York, NY, USA, 2007, pp. 1455–1464. http://dx.doi.org/10.1145/1240624.1240844.
[7] J.F. Pane, B.A. Myers, L.B. Miller, Using HCI techniques to design a more usable programming system, in: IEEE 2002 Symposia on Human Centric Computing Languages and Environments, 2002. Proceedings, 2002, pp. 198–206. http://dx.doi.org/10.1109/HCC.2002.1046372.
[8] M. Berland, V.R. Lee, Collaborative strategic board games as a site for distributed computational thinking, Int. J. Game-Based Learn. 1 (2) (2011) 65–81. http://dx.doi.org/10.4018/ijgbl.2011040105.
[9] N.R. Holbert, U. Wilensky, Racing games for exploring kinematics: a computational thinking approach, in: Proceedings of the 7th International Conference on Games + Learning + Society Conference, ETC Press, Pittsburgh, PA, USA, 2011, pp. 109–118. Retrieved from http://dl.acm.org/citation.cfm?id=2206376.2206390.
[10] C. Kazimoglu, M. Kiernan, L. Bacon, L. MacKinnon, Learning programming at the computational thinking level via digital game-play, Procedia Comput. Sci. 9 (2012) 522–531. http://dx.doi.org/10.1016/j.procs.2012.04.056.
[11] T.Y. Lee, M.L. Mauriello, J. Ingraham, A. Sopan, J. Ahn, B.B. Bederson, CTArcade: learning computational thinking while training virtual characters through game play, in: CHI'12 Extended Abstracts on Human Factors in Computing Systems, ACM, New York, NY, USA, 2012, pp. 2309–2314. http://dx.doi.org/10.1145/2212776.2223794.
[12] H. Kato, A. Ide, 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, L. Erlbaum Associates Inc., Hillsdale, NJ, USA, 1995, pp. 195–199. http://dx.doi.org/10.3115/222020.222171.
[13] J. O'Kelly, J.P. Gibson, RoboCode & problem-based learning: a non-prescriptive approach to teaching programming, SIGCSE Bull. 38 (3) (2006) 217–221. http://dx.doi.org/10.1145/1140123.1140182.
[14] E. Horvitz, Principles of mixed-initiative user interfaces, in: Proceedings of the SIGCHI conference on Human Factors in Computing Systems, ACM, New York, NY, USA, 1999, pp. 159–166. http://dx.doi.org/10.1145/302979.303030.
[15] R.L. Goldstone, J.Y. Son, The transfer of scientific principles using concrete and idealized simulations, J. Learn. Sci. 14 (1) (2005) 69–110.
[16] K. Sugiyama, S. Tagawa, M. Toda, Methods for visual understanding of hierarchical system structures, IEEE Trans. Syst. Man Cybern. 11 (2) (1981) 109–125. http://dx.doi.org/10.1109/TSMC.1981.4308636.
[17] B.G. Glaser, A.L. Strauss, The Discovery of Grounded Theory: Strategies for Qualitative Research, Transaction Publishers, 2009.
[18] C.C. Chase, D.B. Chin, M.A. Oppezzo, D.L. Schwartz, Teachable agents and the protégé effect: increasing the effort towards learning, J. Sci. Educ. Technol. 18 (4) (2009) 334–352. http://dx.doi.org/10.1007/s10956-009-9180-4.
[19] A. McAuley, The MOOC model for digital practice, 2010.