CTArcade: Learning Computational Thinking While Training Virtual Characters Through Game Play

Tak Yeon Lee
Human-Computer Interaction Lab, Department of Computer Science, University of Maryland, College Park, MD 20742 USA
tylee@umd.edu

Awalin Sopan
Human-Computer Interaction Lab, Department of Computer Science, University of Maryland, College Park, MD 20742 USA
awalin@cs.umd.edu

Matthew Louis Mauriello
Human-Computer Interaction Lab, Department of Computer Science, University of Maryland, College Park, MD 20742 USA
mattm@cs.umd.edu

June Ahn
Human-Computer Interaction Lab, College of Information Studies, University of Maryland, College Park, MD 20742 USA
juneahn@umd.edu

John Ingraham
Human-Computer Interaction Lab, Department of Computer Science, University of Maryland, College Park, MD 20742 USA
john.ingraham71@gmail.com

Benjamin B. Bederson
Human-Computer Interaction Lab, Department of Computer Science, University of Maryland, College Park, MD 20742 USA
bederson@cs.umd.edu

CHI 2012, May 5–10, 2012, Austin, TX, USA.
ACM xxx-x-xxxx-xxxx-x/xx/xx.

Abstract
In this paper we describe CTArcade, a web application framework that seeks to engage users through game play resulting in the improvement of computational thinking (CT) skills. Our formative study indicates that CT skills are employed when children are asked to define strategies of common games such as Connect Four. In CTArcade, users can train their own virtual characters while playing games with it. Trained characters then play matches against other virtual characters. Based on reviewing the matches played, users can improve their game character. A basic usability evaluation was performed on the system, which helped to define plans for improving CTArcade and assessing its design goals.

Keywords
Computational Thinking, Learning and Games

ACM Classification Keywords
H5.m. Information interfaces and presentation (e.g., HCI): Miscellaneous.

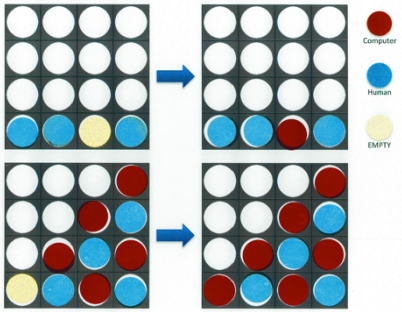

Figure 1 Image of a child's representation of two rules. (Red: computer, Blue: human, and yellow dots: empty cells) The top rule shows that the human can be blocked from winning by playing in an empty cell when the human would otherwise get four in a row. The bottom rule shows how the computer can win the game by completing 4 diagonal cells in a row.

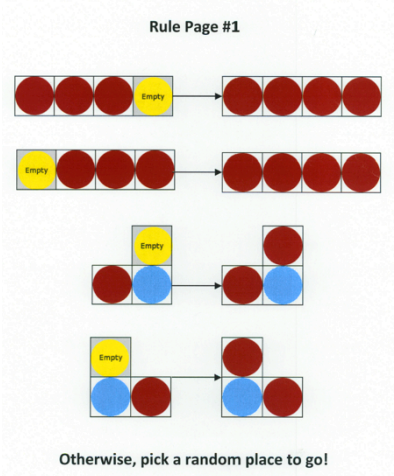

Figure 2 Image of 4 rules created by a child. The computer player had to follow these rules. The top two rules show how to win by completing 4 horizontal cells. The bottom two rules show how to make progress by placing a second cell on a diagonal.

Introduction

One method of building the workforce of tomorrow is to increase the number of students who pursue computer science (CS) during their academic career, which is currently a challenging goal in the United States. Maintaining diversity among students entering CS education remains a serious issue [3]. This has been a trend, indicated by earlier studies that looked at enrollment in secondary and post-secondary computing courses, concluding that the numbers are at an all-time low especially when viewing women and minority populations [17]. Of those students that enroll in CS between 2000 and 2006, 48% become disinterested and end up dropping out due to the extreme difficulty of programming and the students' lack of a proper background from their previous education [12].

From a pedagogical standpoint, CS education is evolving to stress computational thinking (CT) skills earlier in a student's education. Wing [19] asserts that CT is fundamentally about learning how to solve problems through skills such as the abstraction of problems, the definition of appropriate representations, and the development of solutions. Google also promotes a set of four CT skills that it feels are fundamental to CS: Decomposition of problems, Pattern Recognition, Abstraction, and Algorithmic Thinking [6].

In this paper, we present an approach to engage learners into thinking about CT based on their existing game-play strategies. We describe a formative user study that confirmed and expanded our thinking, and then describe CTArcade, a web application platform, that provides scaffolding to help learners think about their thinking while developing their own game algorithms.

Related Work

There have been a number of strategies aimed at making CT more engaging and easy to grasp. Initial efforts focused on using programming such as Logo to allow students to build simulations, robots, and other projects [14]. Subsequent projects, Alice, Scratch, and Agent Sheets [9, 15, 16], focus on creating visual authoring environments for young learners to create animations and video games. Another approach (i.e. Codecademy[1]) is to make traditional syntax learning more engaging with interactive tutorials.

We observe a shared pedagogical trajectory in these programming/authoring environments. Users first learn the primitive syntax of the language as a means to build simple programs that they are interested in. While these approaches are successful at introducing programming to new learners, recent thought in CT suggests that one could start with natural human pursuits and then connect CT to these activities *in situ*. For example, CT skills such as debugging and distributed computation occur naturally while playing collaborative board games, dominoes or racing games such as Mario Kart [2, 8, 11].

In order to embed CT within natural activities, a significant problem is encountered concerning the design of tools that help individuals become cognizant of their intrinsic CT skills – and then translating their natural thinking patterns to a related computational syntax or vocabulary. Researchers find that individuals frequently describe computational ideas such as if-then logic, looping, and iteration in their everyday lives; however, they have extreme trouble translating this

[1] codecademy.org

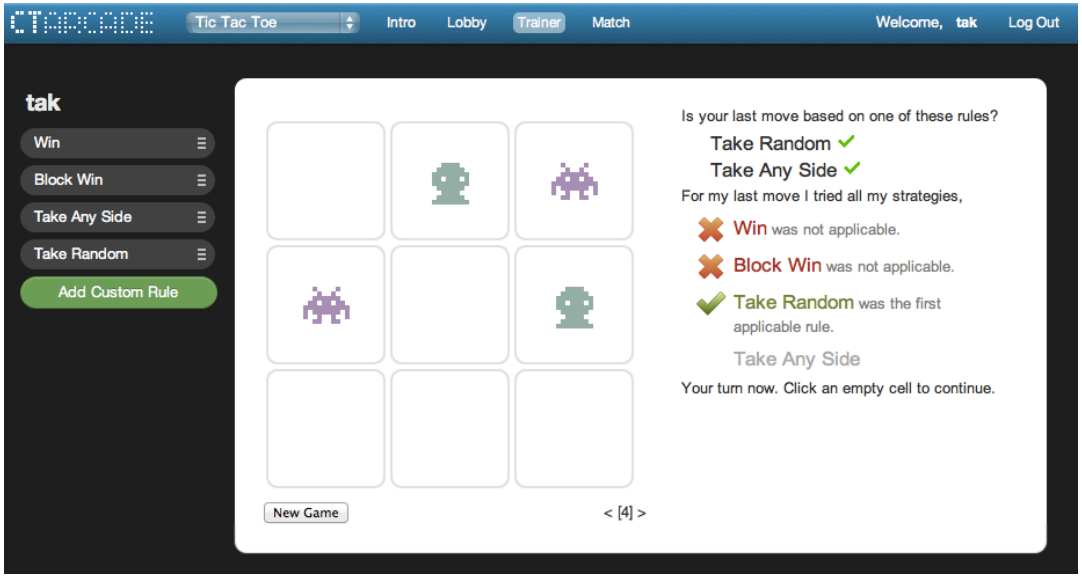

Figure 3 *Trainer* mode. (Left) An ordered list of rules that user's character currently knows. Rules can be moved up & down to re-prioritize. (Center) Tic-Tac-Toe board where users can play with their own character. (Right) Console showing rules that have been applied to the last move made. In this example, user's latest move (middle tile in the upper row) matches with TAKE RANDOM and TAKE ANY SIDE rules.

tacit knowledge into a generalized CS vocabulary or programming conventions [7, 13].

Social factors have been proven to be important in education; however, not many CT educational systems involve social interaction as a main characteristic. Scratch is one example of a basic social environment where children and adult users share their code and improve together. Robocode [12], on the other hand, provides competitive environments where each user programs the behaviors of tank objects to engage other users in an online competition, which forces them to learn and adapt their code to deal with the various scenarios they encounter.

Formative User Study

A formative study of game play of Connect Four was conducted with a team of 7 children, ages 7 - 11, who are part of a participatory design program at the University of Maryland's Human Computer Interaction Lab (HCIL)[2], using methods of Cooperative Inquiry [4]. We asked the children to create game-play strategies using a paper-based representation of the game (Figure 1). After the completion of the previous phase, each pair of children played as *human* and *computer* player respectively – *computer* players were restricted to only play particular rules in the ordered list of the rules (Figure 2).

The study found that the children could easily grasp the rules of the game and were able to verbalize game play strategies. However, when asked to decompose their innate thinking into abstract representations, the children showed great difficulty and confusion. We believe that this inflection point, where **tacit knowledge is abstracted and generalized**, may be a critical place to design game interfaces that help learners to explicitly link their game actions to the abstracted representations and algorithms that describe them.

The next design consideration we learned is **moving from concrete to abstract computational thinking**, as traditional learning theory suggests that children progress from thinking concretely first and then to abstract principles [18]. Following that design consideration, the idea of *concreteness fading* [5] should be applied. Interfaces designed in this way increasingly highlight broader algorithmic strategies while gradually reducing the salience of the specific game situation. Finally, the game design should **minimize the split attention effect** [1] where asking learners to pay attention to, and integrate, separate

[2] http://www.cs.umd.edu/hcil/

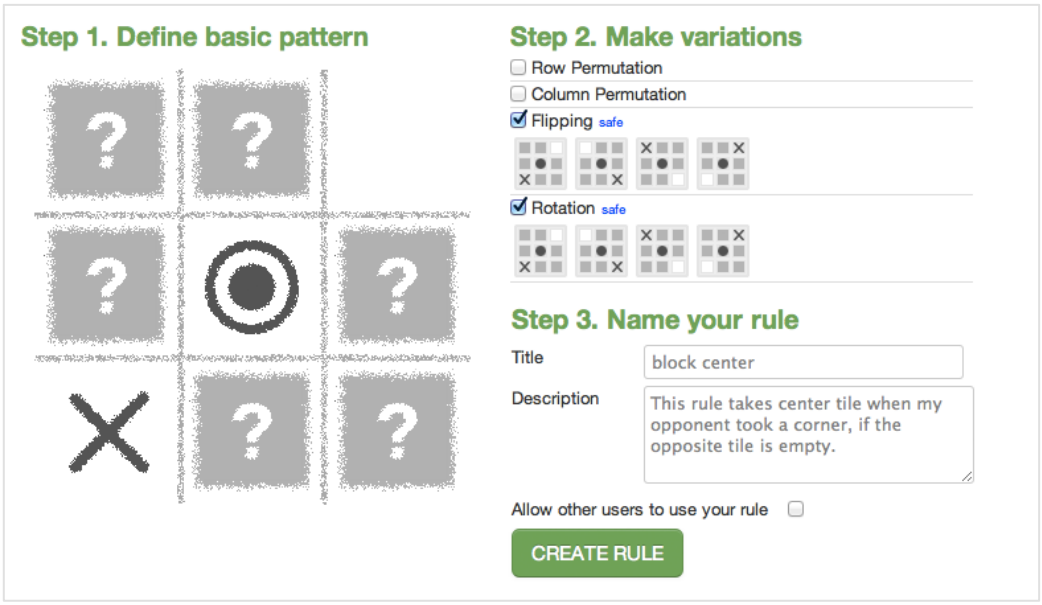

Figure 4 Custom Rule Creator. Defining a basic pattern and generalizing it with various transforming operations can create a new rule.

elements of information leads to increased cognitive load.

CTARCADE

CTArcade has three main components: *Trainer* and *Match Reviewer*. The user's goal is to train his/her own character in *Trainer*, and then win matches against other characters.

In *Trainer* mode, users can extract new rules through game play and teach their characters. How to teach new rules is one of our research questions, and we came up with several ways. First, a user can select one of the predefined rules[3] that match the user's latest move (Figure 3). This feature addresses the first design consideration – help users link their tacit knowledge to the abstracted representations and algorithms.

[3] Six predefined rules are provided. WIN, BLOCK WIN, TAKE CENTER, TAKE ANY SIDE, TAKE ANY CORNER, TAKE OPPOSITE CORNER and TAKE RANDOM.

If the user wants to create a new rule from scratch, the custom rule creator (Figure 4) provides the two-step method – defining a base pattern on the board and then generalizing the pattern with various transforming operations. The method is an example of *concreteness fading*, a design consideration found during the formative study.

After training their own character, users can test it with other characters in *Match Reviewer* mode. Due to the randomness of the Tic-Tac-Toe game just one match is not enough for assessing how well the character is trained. Therefore the Match Reviewer mode runs a predefined number of matches (currently 20) with another character selected by the user from a list and presents the summary of results. To see the effect of minimizing split attention, we tried four types of visualizations; 1) *List view* (Figure 5) simply shows all the games in full detail; 2) *Group by winner* (Figure 6) is useful for focusing on winning/losing games; 3) *Stepwise animation* (Figure 7) is suitable to see temporal trends of all the games; and 4) *Game tree graph* (Figure 8) compresses similar board states into a graph node and connects them with accumulative edges whose thickness represents how often the transition occurred.

DISCUSSION

Four graduate students in computer science participated in preliminary usability testing. This testing consisted of two pairs of students that were asked to participate in a series of exercises. In the first set of exercises, each student was asked to interact with CTArcade on their own. Following the solo activities, participants were then asked to interact with their partner using the *Trainer* and *Match Reviewer*

Figure 5 *List view* simply shows all the states of each game.

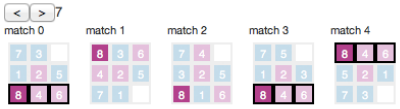

Figure 6 *Stepwise Animation* provides a control to play all the games forward/backward.

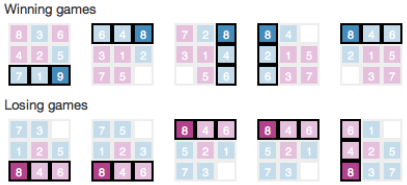

Figure 7 *Group by Winner* summarizes winning/losing games separately.

Figure 8 *Game tree graph* shows patterns of how winning/losing matches branched out.

components. Following these interactions, the participants re-trained their characters. This brief evaluation provided feedback that was immediately used in a general redesign of the application. We believe that CTArcade's Tic-Tac-Toe components are ready for formal assessment concerning its viability as a learning tool.

As part of this formal assessment, we plan to use an interactive survey that is part of the account creation process. The user's system usage will be tracked as a baseline for understanding how they are interacting with CTArcade and to provide a weight for their CT skill improvement level. The formal assessment will also include a post assessment at the end of a specified assessment period. As new components and features are introduced into CTArcade, we plan to continue receiving data and evaluation from previous users and we will also conduct similar pre/post assessments of new users to gauge the impact of our updates.

One of our aspirational research questions is whether social interaction between users can strengthen the effectiveness of learning. We do not yet have our desired level of support for this, so the next step is to add features promoting social interaction such as synchronous game play in which two players can play against each other at the same time or sharing one's custom rules with other players.

Another direction of the project is selecting more games for CTArcade. Although Tic-Tac-Toe was a reasonable choice as the first game, it has several limitations. First, its simplicity and the low ceiling that can be expressed allow matchups between two reasonably good characters to continually end in a draw. Second, its competitive focus is likely to alienate some learners. Thus, we are investigating the creation of much more collaborative games.

CONCLUSION

In this paper we introduced the first implementation of the CTArcade platform. CTArcade has a unique goal: engaging users to learn computational thinking skills while training his/her virtual characters that play with/against other characters. CTArcade also seeks to implement innovative approaches such as training by demonstration and debugging with visualization. Future work falls in three categories: to promote social interaction, increase the variety of games with differing characteristics, and studying its pedagogical effectiveness.

REFERENCES

[1] Ayres, P., and Sweller, J. The split-attention principle in multimedia learning. In *The Cambridge Handbook of Multimedia Learning*, R. E. Mayer, Ed. Cambridge University Press, New York, NY, 135-146.

[2] Berland, M., and Lee, V. R. Collaborative strategic board games as a site for distributed computational thinking. *International Journal of Game-Based Learning 1*, 2 (2011), 65-81.

[3] Chen, X. Students Who Study Science, Technology, Engineering, and Mathematics (STEM) in Postsecondary Education. Technical report, National Center for Education Statistics, Institute for Education Sciences, Washington DC, USA 2009, 2009.

[4] Druin, A. Cooperative inquiry: Developing new technologies for children with children. In *Proc. CHI 1999*, ACM Press (1999), 592-599.

[5] Goldstone, R. L., and Son, J. Y. The transfer of scientific principles using concrete and idealized

simulations. *The Journal of the Learning Sciences 14*, 1, (2005), 69-110

[6] Google. Exploring Computational Thinking. http://www.google.com/edu/computational-thinking/

[7] Gudzial, M. Education paving the way for computational thinking. *Communications of the ACM 51*, 8 (2008), 25-27.

[8] Holbert, N. R., and Wilensky, U. Racing games for exploring kinematics: A computational thinking approach. Paper presented at *AERA 2011*, New Orleans, LA, USA, 2011.

[9] Kelleher, C., Pausch, R., and Kiesler, S. Storytelling alice motivates middle school girls to learn computer programming. In *Proc. CHI 2007*, ACM Press (2007), 1455-1464

[10] Moland, K. J., Decline of U.S. student enrollment in computer science programs, Southeastcon, 2011 Proceedings of IEEE , vol., no., pp.297-299, 17-20 March 2011, doi: 10.1109/SECON.2011.5752953

[11] Nasir, N. S. Individual cognitive restructuring and the sociocultural context: Strategy shifts in the game of dominoes. *The Journal of the Learning Sciences 14*, 1 (2005), 5-34.

[12] O'Kelly, J., and Gibson, J. P., RoboCode & problem-based learning. ACM SIGCSE Bulletin, 38(3):217, June 2006.

[13] Pane, J. F., Ratanamahatana, A., and Myers, B. A. Studying the language and structure in non-programmers' solutions to programming problems. *International Journal of Human-Computer Studies 54* (2001), 237-264.

[14] Papert, S. *Mindstorms: Children, computers, and powerful ideas*. Basic Books, New York, NY, USA, 1993.

[15] Repenning, A., Webb, D., and Ioannidou, A. Scalable game design and the development of a checklist for getting computational thinking into public schools. In *Proc. SIGCSE 2010*, ACM Press (2010), 265-269

[16] Resnick, M., Maloney, J., Monroy-Hernandez, A., Rusk, N., Eastmond, E., Brennan, K., et al. Scratch: Programming for all. *Communications of the ACM 52*, 11 (2009), 60-67

[17] Stephenson, C., Gal-Ezer, J., Haberman, B., & Verno, A., The New Educational Imperative : Improving High School Computer Science Education The New Educational Imperative. *(C. Stephenson, Ed.) Computer, 90. Association for Computing Machinery,* 2005

[18] Uttal, D. H., Liu, L. L., and DeLoache, J. S. Taking a hard look at concreteness: Do concrete objects help young children learn symbolic relations? In *Child Psychology: A Handbook of Contemporary Issues*, Lawrence Balter, Catherine Tamis-Lemonda, Eds., Psychology Press, New York, NY, 177-192. Psychology Press, New York, NY, 2000.

[19] Wing, J. M. Computational thinking. *Comm. of the ACM 49*, 3 (2006), 33-35.