Developing an Educational Game to Support Cognitive Learning

Cagin Kazimoglu, Mary Kiernan, Liz Bacon and Lachlan MacKinnon
University of Greenwich

Introduction

This paper outlines how an educational game can be used to support the learning of programming within the Computer Science (CS) discipline and reports on the qualitative results of a series of rigorous studies of the use of this game by first-year introductory programming students. Although this paper applies to the CS discipline, computational thinking (CT) as an intrinsic part of the games process is applicable to any discipline. This is because CT combines logical thinking with CS concepts to produce a recipe for solving problems, regardless of where a problem lies.

Many studies indicate that learning through educational games appeals widely to students, regardless of their backgrounds (Liu *et al*, 2011; Papastergiou,2009). However, though many of these studies demonstrate enthusiasm for educational games and indicate that games can enhance motivation for learning, they offer very few conclusions about what students learn from playing them or whether or not they acquire cognitive abilities thereby (Denner *et al*, 2012; Connolly *et al*, 2011).

Learning to Program

Introductory programming students often perceive the learning of computer programming as difficult. Guzdial (2011) emphasises that a 30-50% worldwide failure rate in introductory programming courses has been reported for decades. Even after passing their programming courses, many students still do not have the ability to use programming codes to solve problems within the CS discipline (Loftuset al, 2011). One reason for this may lie within the nature of computer programming. Learning to program in order to solve real-life problems successfully requires comprehending abstract concepts about CS and arranging these into a rational order.

Computational Thinking

This term was first introduced by Papert (1996) as a powerful infrastructure for learning. Wing (2006) expanded this notion and argued that CT is a problem-solving approach which combines logical thinking with CS concepts to produce a way to solve problems. It is widely accepted that CT is concerned with conceptualising, developing abstractions and designing solutions, which overlaps with logical thinking and requires fundamental concepts similar to computing (Wing, 2011; Wing, 2008). Although there is still lack of clarity of definition amongst researchers (Berland & Lee, 2011), many agree that there are five key ingredients involved in CT:

1. **Conditional logic** refers to solving problems with logical thinking through using various computational models. Students can evaluate a problem and specify appropriate criteria in order to develop applicable abstractions. At this stage, students distinguish between problems and understand them at an abstract level.

2. **Building algorithms** involves the construction of step-by-step procedures for solving a particular problem and developing abstractions robust enough to be reused to solve similar problems.

3. **Debugging** is the analysis of problems and errors in logic or activities. At this stage, students receive feedback on their algorithms and evaluate them accordingly.

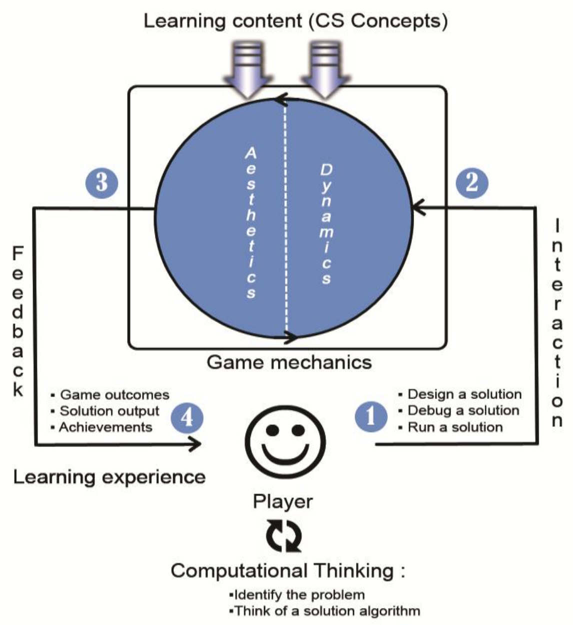
4. **Simulation** is the demonstration of algorithms and involves designing and implementing models on the computer, based on the built algorithm(s). At this stage, students design or run models as test beds, to make decisions about which circumstances to consider when completing their abstraction.

5. **Socialising** refers to coordination, cooperation and/or competition during the stages of problem solving, algorithm building, debugging and simulation. It is reported that socialising is one distinct feature of CT that distinguishes it from traditional computer programming, as this characteristic allows brainstorming, assessment of incidents and strategy development among multiple parties.

Game-Based Learning (GBL)

According to the large survey study undertaken by the Interactive Games Association (2012), the top two reasons why people play games are: a) despite being challenging, playing games is an entertaining activity; b) games provide meaningful feedback that engages and motivates players to continue to play. The survey results also show that many players spend considerable time playing games and they also demonstrate systematic plans to overcome certain challenges during their game-play, even when they do not do well in the game.
As games are immersive environments, it is imperative to harness this energy into learning for educational purposes, particularly in the practice of CT, so that students will be able to transfer knowledge and skills acquired from games to other problems they encounter when learning computer programming (Kumar & Sharwood, 2007).

Figure 1: Interaction - Feedback Loop Model (IFLM) for Games Based Learning

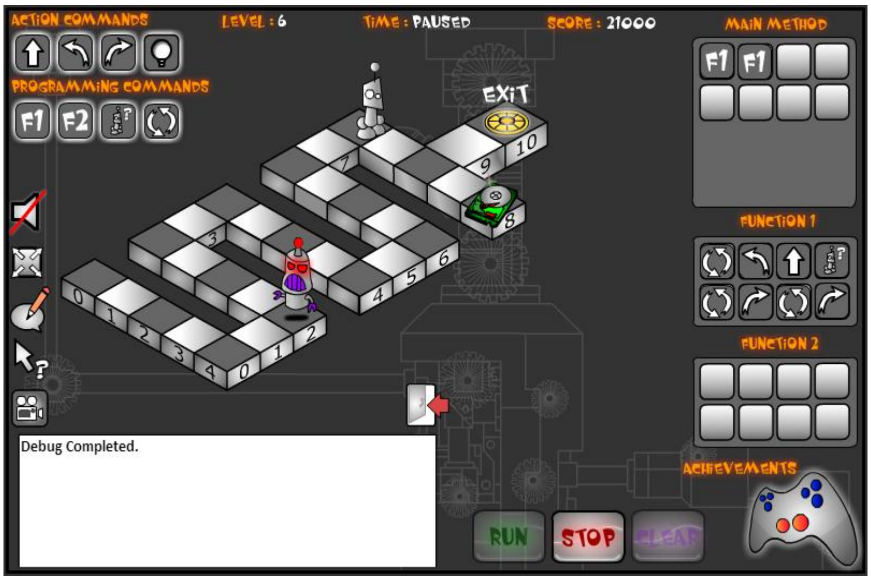


In Figure 1, we developed the *Interaction-Feedback Loop* Model (IFLM) that builds on the work of Garris *et al* (2002) and is proposed as a way to address the flaws of their *input-process-output* model. The crucial difference is that in the IFLM the learning material is an integral part of aesthetics, dynamics and game mechanism, rather than being overlaid on top of the game-play. Thus, we argue that the IFLM was explicitly designed to develop CT skills within a cyclic mechanism and, as players interact within the game and demonstrate good game-play, they also develop their skills in computational thinking through a constructivist approach to learning.

Research Vehicle

In order to test the IFLM*,* a game prototype named *Program Your Robot* (http://www.programyourrobot.com/) (Figure 2) was built, in which the previously-identified CT characteristics (except socialisation, which has still to be fully implemented) were blended into a puzzle-solving game. *Program your robot* was designed to achieve two important goals: firstly, to develop a framework that would allow players to practise their skills and abilities in CT, even with little or no programming background; secondly, to support the learning of procedural and applied knowledge for a limited number of key introductory computer-programming constructs. The theme of the game is to help a robot to escape from a grid platform by reaching the teleport square which will take players to the next level in the game. There are six levels in the current version of the game, each more difficult than the previous.

Figure 2: Program Your Robot game



Debug Completed.

The game provides both *formative* and *summative* feedback to evaluate students' learning progress. Whilst *formative feedback* provides suggestions based on student actions, allowing them to try different solutions and to understand the problem at a deeper level, *summative feedback* rewards students for achieving their goals through an integrated reward system of achievements and high scores.

Associating game-play with Computational Thinking

Four out of the five cognitive skills characterising CT can be practised during the game-play in *Program Your Robot*. The game was not explicitly designed to encourage the remaining CT skill of *socialising* because it was primarily aimed to encourage the development of individual cognitive abilities to support the learning of computer programming. Nevertheless, a limited level of *socialising* can happen indirectly through the reward systems integrated into the game. For those players who want to have additional challenges, a high score list has been designed, to which advanced players can submit their scores and share them with other players. Table 1 shows a set of game activities and describes how students can develop their skills in CT through game-play and, more specifically, through playing *Program Your Robot*.

Table 1: Examples of game activities associated with various categories of CT

Task	Associated CT skill category	Game activity	Rationale of the skill category
Problem identification and decomposition	**Problem Solving**	Help the robot to reach the teleporter. Activate robot's light when robot stands on the teleporter.	CT is described as a problem-solving approach in various studies (Guzdial 2008; Wing, 2006). In conjunction with this, Schell (2008) explains the idea of what a game is: *"a problem-solving activity, approached with a playful attitude."*
Creating efficient and repeatable patterns	**Building Algorithms**	Create a solution algorithm to complete all levels with as few slots as possible. Use functions to create repeatable patterns.	Perkovic *et al* (2010) describe computation as *"the execution of algorithms that go through a series of stages until a final state is reached."*
Practising the debug-mode	**Debugging**	Press the debug button to monitor your solution algorithm to detect any potential errors in your logic.	Wing (2006) describes *"debugging"* as an essential component of both CT and programming.
Practising the run-time mode	**Simulation**	Observe the movements of your robot during the run-time. Can you follow your solution algorithm? Do you observe the expected behaviours?	Moursund (2009) reports that *"the underlying idea in computational thinking is developing models and simulations of problems."*
Brainstorming	**Socialising**	Examine the winning strategies of other players. Compare their solutions with yours. What advice would you give yourself and to them for scoring better in the game? Discuss.	Berland & Lee (2011) refer to the social perspective of CT as *"distributed computation in which different pieces of information or logic are contributed by different players during the process of debugging, simulation or algorithm building."*

Experimental Studies

Two different rigorous studies were designed for first-year introductory programming students, in order to establish a systematic and structured evaluation of *Program Your Robot* and the underlying game model. Over 200 students from two different countries participated in this research and in this paper we share a sample of the qualitative feedback obtained from the studies in relation to the five main characteristics of CT. Student quotes are cited below to demonstrate the flow of game activities relating to the computational thinking stages from the game description.

Associated computational thinking skill: conditional logic

Student 1: *"I tried all sort of tricks using decision making instruction but I failed going any further than level 4 probably because of my poor problem solving skills ☺. Nonetheless, it was good fun crossing the first 3 levels. I liked the fact that the further I was going the more sense it was making."*

Student 2: *"I enjoyed playing the game and it enhanced my knowledge towards methods and how to call declared functions. Overall, I thought the game encourages you to think logically and was really entertaining at the same time."*

Associated computational thinking skill: building algorithm

Student 3: "The game is very well designed and it is one of the games which need a lot of thinking. I got total score of 30750. I didn't experience any errors while finishing this game and it was very easy. In my point of view this game was really good to introduce the fun of programming to students who want to study programming."

Associated computational thinking skill: debugging

Student 4: "I found the debug button useful because it provides messages when I forgot to call a function. However, when I ran the debug mode it didn't find an error or tell me that I have missed the lights or I could not progress until I have done it."

Associated computational thinking skill: simulation

Student 5: *"The game is very well thought out, for example, the demonstration of decision making logic through an if statement was a well thought out example, and the graphical demonstration of this concept is quite creative."*

Student 6: *"I thought that the whole idea behind the game is a good one and I found that using it was quite enjoyable because it included one of the very fundamental premises for teaching programming which is motivating students to continue through regular reward for accomplishment."*

Associated computational thinking skill: socialising

Student 7: *"The game needs a high score page to reward people who use guile and don't rush through the screen. Nonetheless, I enjoyed playing it because I competed against a friend of mine."*

None of the participants stated that they experienced a crash in the game. However, some participants reported bugs (i.e. degraded performance and quality in the game) and almost all of them provided constructive feedback regarding the game mechanics and user interface. Some of these suggestions are cited below:

Student 8: *"It is not clear you need to activate the lights at the end of the run, if you run debug mode it doesn't find an error or tell you that you have missed the lights."*

Student 9: *"The game has an auto save system which is impressive but it doesn't notify* *users of [sic] such a system exist."*

Student 10: *"I have completed all levels in the game. I did not encounter any problems but I found the game interface quite complex and overly done. As the game went on, it became more complex but I managed to understand the concept behind it."*

Conclusion and Future Work

The qualitative feedback gathered from the studies provided strong evidence that *Program Your Robot* has the potential to enhance the computational thinking skills of students who are learning introductory programming. Many participants provided a critical evaluation of the game and their comments provided strong qualitative evidence to support the conclusion that using *Program Your Robot* does provide a motivational route for practising computer-programming constructs and that the progressively more complex levels made them use CT skills to solve the problem. The research presented here is being statistically analysed and quantitative results of three empirical studies will be published in the near future.

Currently, the *Interaction-Feedback Loop Model* (IFLM) has been utilised to develop CT skills within the Computer Science discipline; however, an important area of future work is to ascertain if this model could be used to develop CT skills in students from other disciplines.

Finally, *Program Your Robot* was not designed to measure the social aspect of CT. Possible future work could explore how an explicitly-socialised game-experience could have impact upon students' learning progress. One strategy for doing this would be to adapt *Program Your Robot* into one of the social networks (Facebook, Google+). By this means, the social aspect of learning and how it affects the learning of computer-programming constructs might be investigated at the CT level.

References

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

Connolly, T. M., Stansfield, M., & Hainey, T. (2011). An alternate reality game for language learning: ARGuing for multilingual motivation. *Computers & Education*, 57(1), 1389-1415.

Denner, J., Werner, L., & Ortiz, E. (2012). Computer games created by middle school girls: Can they be used to measure understanding of computer science concepts? *Computers & Education*, 58(1), 240-249.

Garris, R., Ahlers, R., & Driskell, J. E. (2002). Games, motivation, and learning: A research and practice model. *Simulation & gaming*, 33(4), 441-467.

Guzdial, M. (2011). *A Definition of Computational Thinking from Jeannette Wing*. Available at: http://computinged.wordpress.com/2011/03/22/a-definition-of-computational-thinking-from-jeanette-wing/ (last access: October, 2013).

Interactive Games Association, 2012, Available at: www.theesa.com/facts/pdfs/esa_ef_2012.pdf (last access: October, 2013)

Kumar, D. D., & Sherwood, R. D. (2007). Effect of a problem based simulation on the conceptual understanding of undergraduate science education students. *Journal of Science Education and technology*, 16(3), 239-246.

Liu, C. C., Cheng, Y. B., & Huang, C. W. (2011). The effect of simulation games on the learning of computational problem solving. *Computers & Education*, 57(3), 1907-1918.

Loftus, C., Thomas, L., & Zander, C. (2011). Can graduating students design: revisited. In *Proceedings of the 42nd ACM technical symposium on Computer science education*, 105-110. ACM.

Papastergiou, M. (2009). Digital Game-Based Learning in high school Computer Science education: Impact on educational effectiveness and student motivation. *Computers & Education*, 52(1), 1-12.

Papert, S. (1996). An Exploration in the Space of Mathematics Educations, *International Journal of Computers for Mathematical Learning*, Vol. 1, No. 1, 95 – 123.

Werner, L., Campe, S., & Denner, J. (2012). Children learning computer science concepts via Alice game-programming. In *Proceedings of the 43rd ACM technical symposium on Computer Science Education* (pp. 427-432). ACM.

Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, 49(3), 33-35.

Wing, J. M. (2008). Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 366(1881), 3717-3725.

Wing, J. M. (2011). Computational thinking. In *VL/HCC* (p. 3).