A Coder's Game

By: Nsomma Alilonu
Independent Work Seminar 04: Help Future Computer Science Students Learn Computer Science
Project Advisor: Robert Fish

Motivation and Goal:

Despite an increasing fascination with computational devices and applications in modern society, enrollment in the introductory computer science courses of most institutions has been met with an abysmal 20 - 40% dropout rate.[1] Interestingly enough, these rates had little relation to the students' math ability, prior computing experience, or success in school, but more so the students' attitudes and expectations, in addition to educational methods of the university.[2] The role of these more peripheral factors in a novice programmer's experience is further corroborated by another unfortunate statistic: 53.3% of females and 32.6% of males (N = 118) in a survey given to the dropouts of an introductory computer science course listed "Computer science is boring" as one of the reasons they dropped out.[3]

The goal of this project is to make learning computer science fun, specifically through game development. This is important because introducing students to these concepts with an element of fun can encourage them to stay in computer science and find their passion within it. Whether they remain interested in game development, which can lead to leaps in computer vision, data management, and processing, or find themselves drawn to another concentration within the realm of computer science, studying in this field gives them the potential to solve the world's most difficult problems.

Problem Background and Related Work:

I investigated several educational programming games to more deeply understand the approaches to the aforementioned problem already undertaken by other programmers. RobotOn! is one of these games, teaching programming comprehension to the players by having them uncomment or comment lines of code, trace variables, etc. [4] Programming comprehension is essential to the learning process, however, RobotOn! only taught programming comprehension, such that the user had no opportunity to add their own code to the code block. Alice2 had a similar problem, using drag and drop programming as a tool by which the user could create a scene in the application[5]. This prevents syntax errors, which is important as this can discourage novice programmers, but it didn't expose the users to a

[1] Kinnunen, Päivi, and Lauri Malmi, "Why students drop out CS1 course?.", 97.
[2] Kinnunen, Päivi, and Lauri Malmi, "Why students drop out CS1 course?.", 98.
[3] Biggers, Maureen, Anne Brauer, and Tuba Yilmaz, "Student perceptions of computer science: a retention study comparing graduating seniors with cs leavers.", 405.
[4] Miljanovic, Michael A., and Jeremy S. Bradbury, "Robot on! a serious game for improving programming comprehension."
[5] Kelleher, Caitlin, et al., "Alice2: programming without syntax errors."

coding language. Sorceress of Seasons was another game I examined. It provided an excellent progression from easier to more advanced programming concepts and involved user input, but the language used by the users was only pseudocode[6]. In contrast to the previous examples, in *A Coder's Game* the players will type in a mimic IDE filled with code actually used in the program, providing greater relevance to the game itself and exposure to the coding environment.

Approach:

I plan to achieve my goal through the approach of gamification. Gamification has been shown to have many benefits, including increased enrollment, less dropouts, more student participation, and more lecture downloads.[7] A potential downside of gamification is that it can often fail to intrinsically motivate students to learn more about what the game is trying to teach them. This is why I chose to approach the topic of game development through a game, unlike the other approaches in the previous section. The expected users of this game already have an inherent interest in games, so learning about their development is personally relevant to them. This is more likely to avoid the problem of intrinsic motivation, as well as teach them basic introductory concepts of computer programming through the lens of game development.

Plan:

The project will essentially involve two phases, the development of the game environment and the development of the coding environment. The player will play the game, which involves attacking and/or avoiding enemies, for a certain amount of time before receiving a coding challenge (presented as enemy retaliation). The coding challenge will involve the user filling in the blanks in lines of code, receiving hints if they miss an answer a certain amount of times. Once it is solved, a new attack is made available, and the cycle repeats with more difficult coding challenges. Points will be allocated depending on coding accuracy, gaming performance, and time survived. Potential difficulties would be finding an appropriate mimic IDE, ensuring the user can save their progress, and creating a leaderboard. A contingency for the former would involve separate HTML forms with blocks of typed code, and the latter two are stretch goals (as the game can function without them).

Evaluation:

To evaluate this project, I plan to issue the coding challenge aspect of the game to a control group, and the entire game to an experimental group. After completing the challenge, I will issue an assessment to both groups. In this assessment, the users will rate how much they enjoyed learning Javascript, and whether they would like to learn more Javascript in the future. A significant difference between these groups will serve as a measure of success.

[6] Bonner, Desmond Carletus. "A game-based learning approach to increase female participation in science, technology, engineering, and mathematics fields."
[7] Narasareddygari, Mourya Reddy, Gursimran S. Walia, and Alex Radermacher. "Gamification in Computer Science Education: a Systematic Literature Review."

Bibliography:

Biggers, Maureen, Anne Brauer, and Tuba Yilmaz. "Student perceptions of computer science: a retention study comparing graduating seniors with cs leavers." *ACM SIGCSE Bulletin* 40, no. 1 (2008): 402-406.

Bonner, Desmond Carletus. "A game-based learning approach to increase female participation in science, technology, engineering, and mathematics fields." (2015).

Kelleher, Caitlin, Dennis Cosgrove, David Culyba, Clifton Forlines, Jason Pratt, and Randy Pausch. "Alice2: programming without syntax errors." In *User Interface Software and Technology*, vol. 2, no. 2, pp. 35-36. 2002.

Kinnunen, Päivi, and Lauri Malmi. "Why students drop out CS1 course?." In *Proceedings of the second international workshop on Computing education research*, pp. 97-108. 2006.

Miljanovic, Michael A., and Jeremy S. Bradbury. "Robot on! a serious game for improving programming comprehension." In *Proceedings of the 5th International Workshop on Games and Software Engineering*, pp. 33-36. 2016.

Narasareddygari, Mourya Reddy, Gursimran S. Walia, and Alex Radermacher. "Gamification in Computer Science Education: a Systematic Literature Review." In *The 125th ASEE Annual Conference & Exposition.--Computers in Education Division*. 2018.