(Re)Engaging Novice Online Learners in an Educational Programming Game*

Michael J. Lee
Department of Informatics
New Jersey Institute of Technology
Newark, NJ 07102
mjlee@njit.edu

Abstract

Many people are learning programming on their own using various online resources such as educational games. Unfortunately, little is known about how to keep online educational game learners motivated throughout their game play, especially if they become disengaged or frustrated with their task. Keeping online learners engaged is essential for learning programming, as it may have lasting effects on their views and self-efficacy towards computer science. To address this issue, we created a coarse-grained frustration detector that provided users with customized, adaptive feedback to help (re)engage them with the game content. We ran a controlled experiment with 400 participants over the course of 1.5 months, with half of the players playing the original game, and the other half playing the game with the frustration detection and adaptive feedback. We found that the users who received the adaptive feedback when frustrated completed more levels than their counterparts who did not receive this customized feedback. Based on these findings, we believe that adaptive feedback is essential in keeping educational game learners engaged, and propose future work for researchers and designers of online educational games to better support their users.

1 Introduction

Online learning has expanded in popularity with the continued growth of the Internet. Online learning's popularity can be attributed in part to a number of advantages including flexibility, convenience, access, and low cost [9]. Online courses enable students to access materials anytime and anywhere, allowing them to work in an environment of their choice and at their own pace.

However, with the lack of in-person, face-to-face communications in online courses, keeping students motivated to learn can be challenging. Critics have long claimed that online learning is not as effective as traditional classroom learning because of the absence of face-to-face interactions [4]. In a classroom, teachers can gauge students' reactions, body language, and behaviors to determine if and when students are engaged with the course content. Teachers can use these cues to determine the best course of action to re-engage their students. Unfortunately, many of these opportunities to encourage learners go unfulfilled in online contexts (which are essential for any learning setting [1]), negatively affecting learning outcomes [13]. In fact, lack of motivation has been identified as a major cause of the high dropout rates in many online courses [19].

This project explores how feedback, based on users' actions in an online, educational programming game, might affect their motivation to complete more levels. We tested our game with and without a new feedback system with 400 new users, tracking their progress through the game for one week each (approximately 1.5 months total). Our goal was to test if we could successfully detect learners' frustration using models of past users, and how intervening with adaptive feedback might help them re-engage with the content/levels.

2 Related Work

2.1 Dropouts in Introductory Computing Courses

It is well established that introductory programming (CS1) courses in higher education have high dropout rates [11, 15]. A worldwide survey on completion rates reported that only an average of 67% students complete their CS1 course [3]. Further meta-analysis synthesizing 15 years of research on CS1 literature found that the mean pass rate of CS1 courses is 67.7% and that pass rates have not improved over time [3, 23]. Online teaching resources, such as Massive Open Online Courses (MOOCs) fare even worse, with recent numbers reporting fewer than 5% of users completing the curricula they sign up for [10]. Although there are fewer statistics of dropout rates for discretionary online learning settings such as educational games, it is reasonable to presume that rates would be similar, or possibly worse since these online settings lack the mechanisms that compulsory learning resources have to retain their students.

Much of the recent work on engagement in educational programming games has been conducted by our research lab using *Gidget* [16] (www.helpgidget.org). We investigated several strategies for preventing dropout (or *abandonment*), including more personalized error feedback [17] and the inclusion of in-game assessments [18], finding that features that anthropomorphized characters in the game or confirmed understanding could significantly increase engagement [17]. Outside the domain of coding, some researchers have successfully built predictive models of learners' motivational states in similar interactive learning environments [7, 21]. These systems have found predictive success using features related to help seeking, particularly the use of manuals and tooltips.

These and other efforts from prior work have several implications for coding tutorial abandonment prediction. First, many of the most predictive features in prior work have concerned social, instructional, and motivational factors, all of which are difficult to detect using a coding tutorial, especially if the users are using it anonymously. Moreover, the majority of studies have considered dropout at the end of a course of learning, leaving open the possibility that early detection of dropout is not feasible. That said, prior work suggests that some behavioral features, particularly indicators of frustration, may be strong predictors of either engagement or disengagement.

2.2 Detecting Frustration and Providing Feedback

There has been much work in modeling users and providing feedback to change their behavior in the fields of learning science and instructional design. Baker et al.'s work suggests the use of educational data mining and prediction modeling to have educational systems display messages to encourage positive behavior [2]. Rodrigo & Baker identified several coarse-grained measures (e.g., consecutive compilations with the same edit location, the number of pairs of consecutive compilations with the same error, the average time between compilations and the total number of errors) to detect frustration by observing students and analyzing their coding assignment logs [22]. Hattie & Timperley's survey of different kinds of feedback found that the most effective at engaging learners were not those that were related to praise, rewards, or punishment, but rather informative messages relating to feedback about a task and how to do it more effectively [12]. Similarly, Kickmeier-Rust et al. found that adaptive feedback (those relating to a user's current context) helped facilitate users' learning and educational game immersion. However, some researchers report the opposite effects, such as Conati & Manske, who found that adaptive feedback based on learners' actions in an educational game did not lead to learning gains [8]. Our study builds on these previous works, using models of past users' behavior data to detect learners' frustration as a basis to provide an intervention to re-engage them with the content and complete more levels.

3 Method

We modified our free, introductory coding game, *Gidget* (see Figure 1), adding a coarse-grained frustration detector that provided adaptive feedback. The game has a total of 37 levels, where each level teaches a new programming concept (e.g., variable assignment, conditionals, loops, functions, objects) using a Python-like language [16, 18]. For each level, a player has to debug existing code so that the protagonist character can complete its missions. The goal of each level is to pass 1-4 test cases (i.e., statements that evaluate to 'true') upon running the code. After code execution, the game displays which test cases were successful and which ones failed. Each level introduces at least one new programming concept, becoming progressively more difficult as players reach later levels. Therefore, completing more levels means that users are exposed to more programming concepts. Finally, the game also includes a set of help features to help players overcome obstacles while coding on their own [16].

3.1 Modeling Frustration

Based on our literature review and our own prior work using machine learning techniques to detect factors leading to game abandonment [24], we decided to focus on frustration as a primary predictor for addressing disengagement and game abandonment. We were inspired by prior work that found that coarse-grained predictors performed better than fine-grained predictors at detecting frustration [22] and used that model for our frustration detector. As a proof-of-concept, we also decided to to limit the number of factors our disengagement detector distinguished to reduce resource overhead (i.e., client/server processing requirements). We selected a total of five signs of frustration (loosely defined), with the first two adapted from Rodrigo & Baker's work [22] and the latter three adapted from our past work [24]:

1. deviations from the average number of consecutive code executions with the same edit location
2. deviations from the average time between code executions
3. deviations from the average number of code executions
4. deviations from the average time spent on a level
5. deviations from the average time without any activity (idle time)

We defined deviation as values exceeding two standard deviations from the calculated mean of any measure. This value was chosen because two standard deviations away from the mean can be considered "unusual," and we did not want to provide feedback too often (which could result in the *Clippy* effect, where users find the intervention bothersome rather than helpful [20]), or too

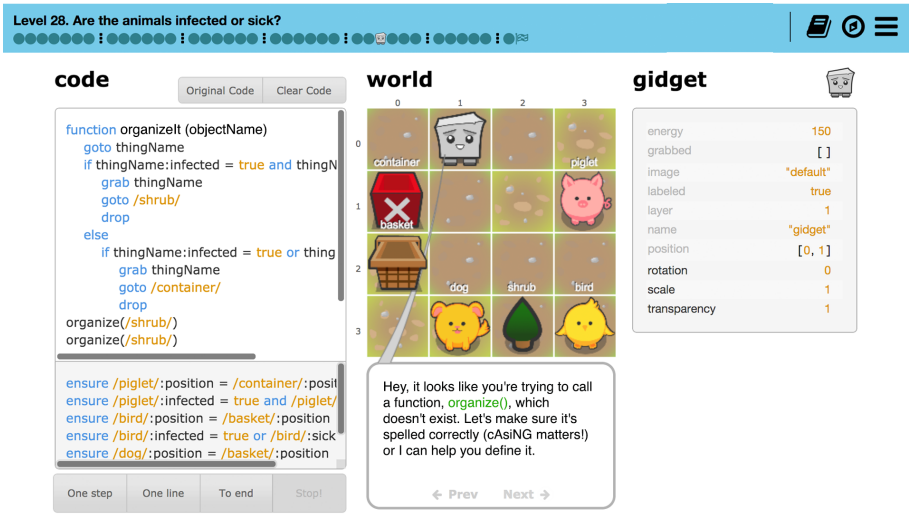

Figure 1: *The experimental condition, displaying an adaptive message (bottom-center speech bubble) that is helping with a function call after detecting consecutive code executions with the same edit location.*

rarely (in which case we miss opportunities to re-engage users). We calculated all of the means and standard deviations for each of the frustration measures above using a data set of 15,448 past users' game logs. These game logs were detailed, including individual players' time spent on level, idle time, all of their code edits, clicks, keystrokes, and execution button usage.

3.2 Adaptive Feedback

We took Hattie & Timperley's [12] and Kickmeier-Rust et al.'s [14] approach in providing users with customized, adaptive feedback relating to their current context (as described in Section 2.2). Our objective was to provide users with contextually relevant information to help (re)engage them with their current task, without giving them exact solutions. To do so, we adapted the design of the Idea Garden, a help system that examines and transforms users' code to provide relevant, but not exact, code examples [5, 6]. For all five cases described above, we used the Idea Garden analysis methodology (described in [5]) with the most recently executed or current version of the users' code to generate a customized, adaptive message for the user (see Figure 1). Longer messages were split into multiple panels that the user could click through (forward and backwards). Finally, because we did not want to directly interrupt the user and allow them to ignore the message if they wanted to, we had the message

generator fade text into the protagonist character's permanent speech bubble at the bottom of the screen. Examples include:

- Hey, it looks like you're trying to call a function, `checkBaskets()`, which doesn't exist. Let's make sure it's spelled correctly (cAsiNG matters!) or I can help you define it.
- You're almost there! The last thing you were working on was on *Line 5*, which seems to be inside a `for` loop block. Remember, `for` loops are written in the following way to iterate through each item in the list:
  ```
  for myPiglet in /piglets/s
      goto myPiglet
  ```
  Don't forget to add tabs for code belonging in the `for` code block!

3.3 Participant Recruitment

We tested our system with a group of 400 online users who were randomly assigned to the regular version of the game (the control condition participants) or a version of the game with the new feedback system (the experimental condition participants) during account creation. The game logged each user's condition so that they would only see the game version they were assigned to, even when coming back to play at a later time. The sign-up page required users to enter their age, gender, e-mail address, state whether they have prior programming experience, and agree/disagree to participating in a research experiment. For the purposes of this study, we only selected users that indicated they were at least 18 years old, had no prior programming experience, and were willing to participate in a research experiment. We set the observation window to 7 days (168 hours) per user to have a consistent timeframe for all users. To expedite the account creation procedure, we did not collect other demographic information such as ethnicity, geographical location, or education level. Participants were required to read and digitally sign an online consent form that briefly described the study. We were intentionally vague in our description of the messages participants might see, stating that we were "testing various types of messages to see how they might help players" to minimize any potential leading or biasing of participants focusing on specific types of messages. However, we e-mailed all participants a copy of the study procedures 7 days after the end of their individual observation window to debrief them, regardless of the condition they were assigned to. Prior to the debrief message (one day after their observation window ended) we sent an e-mail with a link to an optional online questionnaire that asked participants to rate their agreement to the following three statements about their experience with the game on a scale from 1 ('strongly disagree') to 7 ('strongly agree'):

1. The messages that Gidget provided helped me with my goals.

2. The messages that Gidget provided came up too often.
3. The messages that Gidget provided were distracting.

We intentionally under-specified *messages* (and their contents), so that participants from both conditions could interpret what *messages* were on their own. Our system e-mailed two different URLs containing the same questions to their respective participants to distinguish responses between the conditions.

4 Results & Discussion

We provide quantitative results comparing the outcomes from our three groups using nonparametric Chi-Squared and Wilcoxon rank sums tests with $\alpha = 0.05$ confidence, as our data were not normally distributed. Our study was a between-subjects design, with an even split of 200 participants in the control condition group (aged 18-54; median 20), and 200 participants in the experimental condition group (aged 18-55; median 20). Comparisons of demographic data revealed that there were no significant differences between the control and experimental conditions by age or gender (107 males, 88 females, and 5 other or decline to state; and 102 males, 90 females, and 8 other or decline to state, respectively). The key dependent variable in our study was engagement, which we operationalized as the number of levels completed. We also examine the participants' responses to the optional questionnaire.

4.1 Experimental Condition Participants Complete More Levels

All participants completed at least four levels. The range of levels completed in the control and experimental conditions were 4-37 (median 10) and 4-37 (median 13), respectively. We verified that all participants in the experimental condition saw messages from the new feedback system throughout their time playing the game (with more occurring in later, more difficult stages). There was a significant difference in the number of levels participants completed between the two conditions ($W = 42385, Z = 1.986, p < .05$), with the experimental group participants completing more levels.

Since all participants were novice programmers, these results suggest that something about interacting with the new feedback system (frustration detector and adaptive message generator), had a significant positive effect on the experimental condition participants' engagement and ability to complete more levels in the game compared to the control condition participants.

4.2 Unable to Compare Differences in Play Times

Next, we had planned to measure the differences in how long participants took to complete the levels they passed. However, because everyone completed a

different number of levels and the range of completion times for each level were vastly different, we would only able to compare the levels that all 400 participants completed (i.e., Levels 1-4) to see if there were any differences in play times. However, we found that only a few (11 out of 200) participants in the experimental condition received at least one of the new feedback system messages during the first four levels. Therefore, we were unable to compare the differences between the control and experimental group play times since the majority of the experimental group (189/200, or 94.5%) did not experience anything differently from the control group for these common completed levels.

4.3 Experimental Group Agrees Messages Helped with Goals

Finally, we compared our optional questionnaire responses, which had a total response rate of 10.25%, (19 control, 22 experimental). For our analyses, we flipped the scales for Questions 2 and 3 since the statements were negative.

For Questions 1 and 2, our median scores for both conditions were 6 (range 4-7) and 4 (range 2-6), respectively. For Question 3, the control and experimental conditions were 3 and 4 (range 2-6), respectively. Additionally, we found a significant difference between the control and experimental groups agreement to Question 1 ($\chi^2(3, N = 41) = 8.299, p < .05$). However, we did not find significant differences between conditions for Question 2 ($\chi^2(4, N = 41) = 2.410, n.s.$) or Question 3 ($\chi^2(4, N = 41) = 1.385, n.s.$)

We had not expected to find significant differences in our questions because of the low response rate and were excited to find that the experimental group users reported that their messages helped them with their goals—which was the aim of this study. However, we need to explore this result further in future work, as we did not specify exactly which messages in our questions.

5 Conclusion

Our findings show that adaptive feedback messages, triggered by a frustration detector using coarse measures, can significantly improve users' performance in an educational game. In our study, our experimental group participants (those with the frustration detector and adaptive feedback messages) completed significantly more levels than their control group counterparts (who played the game without these additional features). Researchers and educators for online resources for teaching programming may benefit from adding these types of frustration detection and adaptive feedback to their systems.

We have several limitations to our study. First, we recruited participants who opted into a research study. These types of participants may already have high motivation, and therefore may not be completely representative of the larger population. However, we found that the participants in our two groups

were similar to each other, with no significant differences by age or gender. Second, participants from both groups completed a different number of levels, making it impossible for us to compare their usage of the new feedback system (especially because the frustration detector was triggered more often in later levels, which many participants did not reach). Third, we had a low questionnaire response rate in comparison to our full participant pool, which may limit the generalizability of the findings. Future studies may ask participants to complete all levels and everyone to fill out the questionnaire. Finally, we also plan to measure learning outcomes (using pre-post tests) to determine how this feedback system affects learners' knowledge.

Our results from this proof-of-concept study shows that adaptive feedback, triggered by coarse measures to detect frustration, are sufficient in increasing online learners' performance. Our future work will examine these outcomes in more detail to determine what exactly is causing these effects, along with additional coarse frustration predictors, and possibly some fine-grained predictors.

6 Acknowledgements

This work was supported in part by the National Science Foundation (NSF) under grants DRL-1837489 and IIS-1657160. Any opinions, findings, conclusions or recommendations are those of the authors and do not necessarily reflect the views of the NSF or other parties.

References

[1] Susan A Ambrose, Michael W Bridges, Michele DiPietro, Marsha C Lovett, and Marie K Norman. *How learning works: Seven research-based principles for smart teaching*. John Wiley & Sons, 2010.

[2] Ryan Shaun Baker and Paul Salvador Inventado. Educational data mining and learning analytics. In *Learning Analytics*, pages 61–75. Springer, 2014.

[3] Jens Bennedsen and Michael E Caspersen. Failure rates in introductory programming. *ACM SIGCSE Bulletin*, 39(2):32–36, 2007.

[4] Mark Bullen. Participation and critical thinking in online university distance education. *Int. Journal of E-Learning & Distance Education*, 13(2):1–32, 2007.

[5] Jill C Cao. Helping end-user programmers help themselves: the idea garden approach. *Oregon State University, Corvallis, OR*, 2013.

[6] Jill C Cao, Scott D Fleming, Margaret Burnett, and Christopher Scaffidi. Idea garden: Situated support for problem solving by end-user programmers. *Interacting with Computers*, 27(6):640–660, 2014.

[7] Mihaela Cocea and Stephan Weibelzahl. Eliciting motivation knowledge from log files towards motivation diagnosis for adaptive systems. In *International Conference on User Modeling*, pages 197–206. Springer, 2007.

[8] Cristina Conati and Micheline Manske. Evaluating adaptive feedback in an educational computer game. In *International Workshop on Intelligent Virtual Agents*, pages 146–158. Springer, 2009.

[9] Simone CO Conceição. Faculty lived experiences in the online environment. *Adult Education Quarterly*, 57(1):26–45, 2006.

[10] Wenzheng Feng, Jie Tang, and Tracy Xiao Liu. Understanding dropouts in moocs. *Association for the Advancement of AI*, 2019.

[11] Mark Guzdial and Elliot Soloway. Teaching the nintendo generation to program. *Communications of the ACM*, 45(4):17–21, 2002.

[12] John Hattie and Helen Timperley. The power of feedback. *Review of Educational Research*, 77(1):81–112, 2007.

[13] Starr Roxanne Hiltz. Collaborative learning in asynchronous learning networks: Building learning communities. 1998.

[14] Michael D Kickmeier-Rust, Birgit Marte, SB Linek, Tiphaine Lalonde, and Dietrich Albert. The effects of individualized feedback in digital educational games. In *European Conference on Games Based Learning*, pages 227–236. Academic Publishing Limited, 2008.

[15] Päivi Kinnunen and Lauri Malmi. Why students drop out cs1 course? In *ACM ICER*, pages 97–108, 2006.

[16] Michael J Lee. Teaching and engaging with debugging puzzles. *University of Washington, Seattle, WA*, 2015.

[17] Michael J Lee and Amy J Ko. Personifying programming tool feedback improves novice programmers' learning. In *ACM ICER*, pages 109–116, 2011.

[18] Michael J Lee, Amy J Ko, and Irwin Kwan. In-game assessments increase novice programmers' engagement and level completion speed. In *ACM ICER*, 2013.

[19] Lin Y Muilenburg and Zane L Berge. Student barriers to online learning: A factor analytic study. *Distance education*, 26(1):29–48, 2005.

[20] Emerson Murphy-Hill and Gail C Murphy. Recommendation delivery. In *Recommendation Systems in Software Engineering*, pages 223–242. Springer, 2014.

[21] Lei Qu and W Lewis Johnson. Detecting the learner's motivational states in an interactive learning environment. In *AI in Education: Supporting Learning through Intelligent and Socially Informed Tech.*, pages 547–554. IOS Press, 2005.

[22] Ma MT Rodrigo and Ryan S Baker. Coarse-grained detection of student frustration in an introductory programming course. In *ACM ICER*, 2009.

[23] Christopher Watson and Frederick WB Li. Failure rates in introductory programming revisited. In *ACM ITiCSE*, pages 39–44, 2014.

[24] An Yan, Michael J Lee, and Amy J Ko. Predicting abandonment in online coding tutorials. In *IEEE VL/HCC*, pages 191–199, 2017.