Informing the Design of a Game-Based Learning Environment for Computer Science: A Pilot Study on Engagement and Collaborative Dialogue

Fernando J. Rodríguez, Natalie D. Kerby, Kristy Elizabeth Boyer

Department of Computer Science, North Carolina State University, Raleigh, NC 27695
{fjrodri3,ndkerby,keboyer}@ncsu.edu

Abstract. Game-based learning environments hold great promise for supporting computer science learning. The ENGAGE project is building a game-based learning environment for middle school computational thinking and computer science principles, situated within mathematics and science curricula. This paper reports on a pilot study of the ENGAGE curriculum and gameplay elements, in which pairs of middle school students collaborated to solve game-based computer science problems. Their collaborative behaviors and dialogue were recorded with video cameras. The analysis reported here focuses on nonverbal indicators of disengagement during the collaborative problem solving, and explores the dialogue moves used by a more engaged learner to repair a partner's disengagement. Finally, we discuss the implications of these findings for designing a game-based learning environment that supports collaboration for computer science.

Keywords: Engagement, Collaboration, Dialogue, Game-Based Learning.

1 Introduction

Supporting engagement within computer science (CS) education is a central challenge for designers of CS learning environments. More broadly, engagement is a subject of increasing attention within the AI in Education community. A growing body of empirical findings has revealed the importance of supporting learner engagement. Particular forms of disengagement have been associated with decreased learning, both overall and with respect to local learning outcomes within spoken dialogue tutoring systems [1, 2]. Targeted interventions can positively impact engagement; for example, metacognitive support may influence students to spend more time on subsequent problems, and integrating student performance measures into a tutoring system allows them to reflect on their overall performance [3]. A promising approach to support engagement involves adding game elements to intelligent tutors or other learning environments [4, 5] or creating game-based learning environments with engaging narratives [6]; both approaches have been shown to increase student performance and enjoyment in general. However, even with these effective systems, some disengaged

behaviors are negatively associated with learning, and the relationships between engagement and learning are not fully understood.

Collaboration is another promising approach for supporting engagement and can be combined with game-based learning environments [7]. Results have demonstrated the importance of well-timed help for collaborators [8] and the promise of pedagogical agents that support self-explanation [9]. This study considers collaboration in the problem-solving domain of computer science, where a combination of hints and collaboration support may be particularly helpful [10]. However, many questions remain regarding the best sources and types of engagement support in this context.

Game-based learning environments for teaching computer science have started to become popular in recent years. The CodeSpells game [11] aims to teach middle school students how to program in the Java programming language. The ENGAGE project aims to develop, implement, and evaluate a narrative-centered, game-based learning environment that will be deployed in middle school for teaching computer science principles. The game is designed to be played collaboratively by pairs of students. Presently, the project is in its design and implementation phase, conducting iterative refinement and piloting of curriculum and gameplay elements. During this process, we aim to extract valuable lessons about how middle school students collaborate to solve computer science problems and how this collaboration can be supported within an intelligent game-based learning environment.

This paper reports on a pilot study of the ENGAGE curriculum and simulated gameplay elements. In this study, pairs of students collaborated and their collaborative behaviors and dialogue were recorded with video cameras. Nonverbal indicators of disengagement were annotated manually across the videos. We report an analysis of these disengagement behaviors by students' collaborative role, and explore the dialogue moves used by a more engaged learner to repair a partner's disengagement.

2 ENGAGE Game Based Learning Environment

The main goals of the ENGAGE project, which is currently in its design and implementation phase, are to create a highly engaging educational tool for teaching computer science to middle school students, contribute to research on the effectiveness of game-based learning, and investigate its potential to broaden participation of underrepresented groups in computer science. During the first year of the project, the first draft of the curriculum to be used within the environment was developed. The curriculum is based on the CS Principles course under development by the College Board [12] with the goal of shifting focus from a specific programming language (Java, in the case of the existing AP Computer Science course) to the broader picture of computer science concepts. The CS Principles curriculum emphasizes seven *big ideas*:

Table 1. CS Principles focused evidence statement examples

CS Principles Number	Evidence Statement
6b	Explanation of how number bases, including binary and decimal, are used for reasoning about digital data
13a	Explanation of how computer programs are used to process information to gain insight and knowledge
18b	Explanation of how an algorithm is represented in natural language, pseudo-code, or a visual or textual programming language
24a	Use of an iterative process to develop a correct program
30c	Explanation of how cryptography is essential to many models of cybersecurity

1. Computer science is a creative process
2. Abstractions can reduce unimportant details and focus on relevant ones
3. Big data can be analyzed using various techniques in order to create a new understanding or refine existing knowledge
4. Algorithms are a sequence of steps used to solve a problem and can be applied to structurally similar problems
5. These algorithms can be automated using a programming language
6. The Internet has revolutionized communication and collaboration
7. Computer science has an impact on the entire world

Through an iterative process, we selected a subset of the CS Principles curriculum by analyzing the evidence statements [13] for suitability within a game-based learning environment and for appropriateness for the middle school audience. Additional validation of this curriculum will be undertaken by middle school teachers during an upcoming summer institute and through pilot testing. An example of learning objectives to be implemented as game-based learning activities is shown in Table 1.

The setting of the game is an underwater research facility that has been taken over by a rogue scientist. Students take on the role of a computer scientist sent to investigate the situation, reconnect the station's network, and thwart the villain's plot by solving various computer science puzzles in the form of programming tasks. There are two main gameplay mechanics: players can move around in the 3D environment in a similar manner to many 3D platforming games (Fig. 1a), and different devices within the environment can be programmed using a visual programming interface. Players can drag "blocks" that represent programming functions and stack them together to create a program (Fig. 1b).[1] By programming these devices in certain ways, players can manipulate the environment and solve each in-game area's puzzle and move on to the next task. The game sections are divided into four main levels:

[1] This drag-and-drop programming language with blocks is closely modeled after and inspired by Scratch [21], but for compatibility reasons, a customized programming environment is being created for the ENGAGE game.

- *Tutorial*: Students are introduced to the game environment and shown how to use the controls for both gameplay mechanics. They are also given an overview of basic programming concepts (sequences of statements, loops, and conditionals), as well as the concept of broadcasting (sending signals from one device to another).
- *Digital World*: The puzzles in this level involve binary numbers, and students must convert these binary numbers into an understandable form (decimal, text, color image, etc.) in order to solve them and progress. The conceptual objective of this level is that computers communicate in binary and that the meaning behind a binary sequence depends on its interpretation.
- *Cybersecurity*: Before the students can reconnect the station's network, they must establish proper cybersecurity measures so that their communications are not compromised by the villain. In this level, students learn about cryptography and various encryption techniques in order to ensure a safe network connection.
- *Big Data*: The research station that students must restore had been studying different aspects of the undersea environment, including the pollution of the water and how it has affected the life forms that inhabit the area. Students must try to reason about this data by performing basic analyses and creating visual models that are embodied in the 3D environment, which will enable students to progress to the final level.

a) Game environment b) Programming interface

Fig. 1. Engage screenshots

3 Pilot Study

The pilot study was conducted within a computer science elective course for middle school students (ages 11 to 14) at a charter middle school. Students attended 4 sessions lasting between 90 and 120 minutes long, facilitated by members of the ENGAGE project team. Each of the sessions resulted in a corpus of data, though only one of these, which involves solving a binary puzzle within a visual programming environment, is analyzed within this paper. This paper focuses on the final session of the course, in which students worked in pairs to solve three game-based tasks using Build Your Own Blocks, a drag-and-drop visual programming language [14]. This activity simulates programming within ENGAGE, whose programming environment is still under development. Student participants included 18 males and 2 females, though

the female pair was absent on the day the present corpus was collected. This gender disparity is an intrinsic problem in many technology electives and is an important consideration of the ENGAGE project, which will examine differential outcomes for students from underrepresented groups using the game-based learning environment.

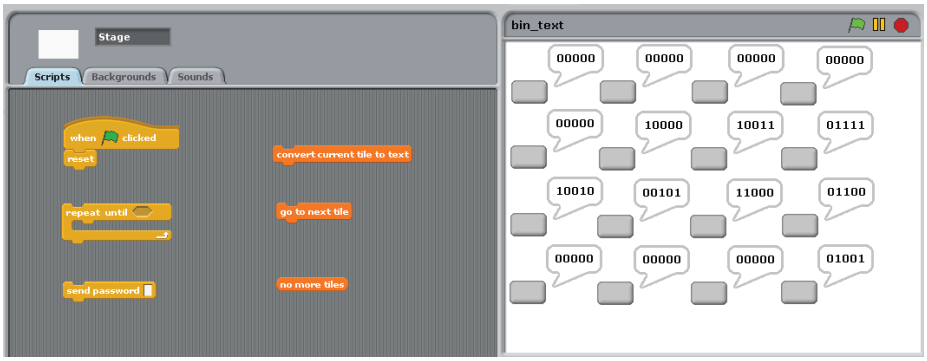

Fig. 2. Visual programming interface for final problem in pilot study

The exercises the students solved involved converting binary numbers into decimal numbers and textual characters. The first problem asked the students to write a program that would convert the given binary numbers into decimal numbers and highlight the cells that contained even numbers. The next problem was identical to the first except that the students had to highlight the prime numbers instead. The final and most challenging problem asked students to convert the binary numbers into textual characters, manually decipher a password, and input this password into their program (Fig. 2). All subroutines corresponding to binary conversions, highlighting blocks, number comparisons, and password entry were provided as blocks that students could use within their own programs. This design choice was made to abstract some complex implementation from students so they could focus on planning and implementing the steps to solve their problem.

For the duration of the exercise, students collaborated in pairs and took turns controlling the keyboard and mouse on a single computer [7]. In computer science education, this paradigm is referred to as pair programming: the *driver* actively creates the solution, while the *navigator* provides feedback [15] (Fig. 3). Research has shown that pair programming can provide many benefits to college-level students taking introductory programming courses, especially those with little to no prior programming experience. A review by Preston [16] highlights some benefits: students create higher-quality programs as a result of the communication of ideas between partners; they can achieve a better understanding of programming by supporting each other through the exercise; and although the activity is collaborative, individual test scores and course performance are also improved.

Students were asked by a researcher to switch roles every six minutes. When a pair would finish a task, they were asked to raise their hands and wait for a researcher to verify their program solution. If it was correct, the researcher would verbally describe the next exercise and set up the programming interface; if not, the researcher would

provide general feedback on the proposed solution. Students were also allowed to raise their hands if they had any questions for the researcher regarding the programming task. The allotted time to complete all three programming tasks was 40 minutes, with two pairs completing them sooner.

4 Video Corpus and Disengagement Annotation

During this pilot study, video was recorded for all nine pairs of students using a tripod-mounted digital camera recording at 640x480 resolution and 30 frames per second. The nine videos were divided into 5-minute segments to facilitate annotation and analysis. Of the total 65 segments, 25 were randomly selected for annotation and serve as the basis for the results presented here (a subset was necessary due to the time requirement of manual annotation, in this case approximately 8 minutes per minute of video). Each segment was manually annotated by a judge for student disengagement by observing for one of three signs of disengagement. First, *posture* was considered to indicate disengagement when gross postural shifts clearly suggested that the student was attending to something other than the programming task; this exaggerated disengaged posture was often accompanied by other indications of disengagement such as off-task speech (Fig. 3b). Another indicator of disengagement was averted *gaze*, which commonly accompanied the other two signs but could occur independently. Finally, students would sometimes engage in off-task *dialogue* with their partners, or even with other students in the classroom. It is important to note that we do not equate off-task behavior with disengagement; there were instances in which students continued to work on the learning task while holding off-task conversations with their partners. Sabourin et al. [17] show that off-task behavior can be a way for students to cope with negative emotions, such as confusion or frustration. Likewise, student disengagement does not necessarily imply off-task behavior. Disengagement in this context is defined primarily as focusing attention on something other than the learning task; identifying cognitive and affective states underlying the disengagement is left to future analyses.

To annotate the videos, each human judge would watch until disengagement was observed by either of the two collaborators, paused the video, annotated the start time of the disengagement event, then continued and annotated the end time, returning to previous points of the video as needed. Judges thus marked episodes of disengagement, as well as who appeared to facilitate re-engagement: did the student shift her attention back to the programming task by herself?; did the student's partner ask for her assistance?; or did an instructor need to arrive to provide feedback or clarify any questions? In order to establish reliability of this annotation scheme, 12 of the 65 video segments were randomly selected and assigned to two judges, and the tagged segments were discretized into one-second intervals in which each student was classified as either engaged or disengaged by each judge. The Kappa for disengagement was 0.59 (87.25% agreement). In other words, approximately 87.25% of the time, both judges applied the same engagement tag. Adjusting for chance (that is, students were more likely to be engaged than disengaged at a given point) the

Kappa agreement statistic was 0.59, indicating fair agreement [18]. For the events on which both judges agreed that disengagement had occurred, the tag for who facilitated re-engagement resulted in 78.57% agreement, or a Kappa of 0.60, indicating moderate agreement.

a) Engaged pair b) Disengaged pair

Fig. 3. Collaborative setup

5 Results

Overall, student drivers spent an average of 16.4% of their time disengaged (st. dev.=16.6%), compared to a much higher 42.6% for navigators (st. dev.=24.1%). This is not surprising, since drivers are more actively engaged in the programming activity. Across both roles, out of the student re-engagement events, 76.8% were annotated as self re-engagements, with the remaining events corresponding to an external source of re-engagement (either partner or instructor). However, the collaborative role plays an important part in self re-engagement: drivers had an 87.7% probability of self re-engaging, while navigators had a lower 68.7% probability of self re-engaging. These findings may indicate that repairing one's own disengaged state is more challenging for the collaborative partner who is not actively at the controls.

We examine instances in which annotators marked that the driver re-engaged a disengaged navigator through dialogue. There are 22 such instances. Four are questions addressed to the collaborative partner, such as, "OK, now where?" and "Do we delete this?" These questions re-engaged the navigator in part because attending to the speaker's questions is a social dialogue norm. Two utterances served as exclamations, e.g., "What the heck?" In these cases, the driver was expressing surprise with an event in the learning environment, which drew the disengaged student's attention back to the task. The remaining utterances were fragments, such as, "Pick up current tile…," though one utterance explicitly reminded the disengaged student about short time remaining, "So we only have a couple of minutes."

To examine these re-engagement events in context, two excerpts are considered (Table 2). In Excerpt A, the navigator gets stuck and raises his hand to ask for help when he notices the instructor is nearby, briefly becoming disengaged while his

partner continues to work on the exercise. The driver then turns to his partner and asks for feedback, causing the navigator to re-engage into the learning activity. In Excerpt B, the students had just received feedback on from the instructor when the navigator engages in off-topic dialogue with another team. Meanwhile, the driver makes a plan and then calls for the navigator's attention, re-engaging him.

Table 2. Dialogue excerpts featuring navigator disengagement

Timestamp	Role	Dialogue Excerpt A
19:25	**Navigator:**	OK, if prime, number is prime. Dang!
		[*Navigator notices instructor nearby, raises hand*]
19:34	**Navigator:**	Uh...
Disengaged		[*Navigator looks away from screen, leans back on seat*]
19:38	**Driver:**	OK, now where?
Re-engaged		[*Navigator points at program block*]
19:40	**Navigator:**	Put it there.
		Dialogue Excerpt B
		[Note: students are discussing '@' symbols]
26:01	**Navigator:**	OK, @'s. Do you want more @'s... (inaudible)
26:08	**Driver:**	One two three four five
Disengaged		[*Navigator looks away to talk to another student*]
26:14	**Driver:**	I have an idea. You (taps navigator's shoulder)
Re-engaged		
26:16	**Navigator:**	Me?

6 Discussion

These excerpts suggest that within a collaborative game-based learning environment for computer science, providing both students with a sense of control may be particularly important. Because it may be more difficult to stay engaged on a task if one is not actively participating in it, particularly for younger audiences, the issue of mutual participation is paramount within the learning environment. The narrative game-based learning framework may prove particularly suitable for addressing this challenge: drivers and navigators can be provided with separate responsibilities and even with complementary information so that the participation of both students is required to complete the game-based tasks. Examples include designing the algorithmic solution to the problem or performing some calculations relevant to the main task; Williams and Kessler [19] state that 90% of students surveyed about pair programming listed these as the tasks that the navigator typically assists with. These may, in turn, help the navigator experience a heightened sense of control, and thereby, engagement.

This pilot study demonstrated that because of strong social norms associated with human dialogue, strategic moves by a partner can serve to re-engage a student. Typically, drivers will ask their partners for feedback if they are unsure of their solution or if they are inexperienced programmers. In these cases, an active conversation between both students occurs, and both students are engaged. An intelligent game-based learning environment that senses disengagement may be able

to scaffold this type of dialogue in order to mitigate disengagement on the part of either student.

7 Conclusion and Future Work

Game-based learning environments hold great promise for supporting computer science education. The ENGAGE project is developing a game-based learning environment for middle school computer science, and we have presented results from an early pilot study for the curriculum and some simulated elements of gameplay in which students worked collaboratively in pairs to solve computer science problems. The results suggest that supporting engagement may be particularly important within a collaborative situation for the student who is not at the controls. Providing both students with an active role during gameplay, and scaffolding dialogue to re-engage a student who has become disengaged, are highly promising directions for intelligent game-based learning environments. Both of these interventions would be well supported within a narrative-centered, game-based learning environment framework.

There are several important directions for future work regarding engagement within game-based learning environments for computer science. First, the current study was very limited in sample size and diversity of participants, so expanding the scope of students considered is a key consideration. It is also important to examine the duration of engagement once re-engagement has occurred and the effectiveness of interventions with respect to longer-term engagement. Additionally, in contrast to the fully manual video annotation presented here, it would be beneficial to integrate automated methods of measuring disengagement, such as the ones presented by Arroyo and colleagues [20]. Finally, addressing issues of diversity and groupwise differences of re-engagement strategies is an essential direction in order to develop game-based learning environments that support engagement and effective learning for all students.

Acknowledgements. This paper relies on the contributions of the entire ENGAGE research team, including James Lester, Bradford Mott, Eric Wiebe, Rebecca Hardy, Wookhee Min, Kirby Culbertson, and Marc Russo. The authors wish to thank Joseph Grafsgaard and Alexandria Vail for their helpful contributions. This work is supported in part by NSF through grants CNS-1138497 and CNS-1042468. Any opinions, findings, conclusions, or recommendations expressed in this report are those of the participants and do not necessarily represent the official views, opinions, or policy of the National Science Foundation.

References

1. Forbes-Riley, K., Litman, D.: When Does Disengagement Correlate with Learning in Spoken Dialog Computer Tutoring? Proceedings of AIED. pp. 81–89 (2011).
2. Cocea, M., Hershkovitz, A., Baker, R.S.J.: The Impact of Off-Task and Gaming Behaviors on Learning: Immediate or Aggregate? Proceedings of AIED. pp. 507–514 (2009).

3. Arroyo, I., Ferguson, K., Johns, J., Dragon, T., Meheranian, H., Fisher, D., Barto, A., Mahadevan, S., Woolf, B.P.: Repairing Disengagement With Non-Invasive Interventions. Proceedings of AIED. pp. 195–202 (2007).
4. Jackson, G.T., Dempsey, K.B., McNamara, D.S.: Short and Long Term Benefits of Enjoyment and Learning within a Serious Game. Proceedings of AIED. pp. 139–146 (2011).
5. Rai, D., Beck, J.E.: Math Learning Environment with Game-Like Elements: An Incremental Approach for Enhancing Student Engagement and Learning Effectiveness. Proceedings of ITS. pp. 90–100 (2012).
6. Rowe, J., Shores, L., Mott, B., Lester, J.C.: Integrating Learning, Problem Solving, and Engagement in Narrative-Centered Learning Environments. *IJAIED*. 21, 115–133 (2011).
7. Meluso, A., Zheng, M., Spires, H.A., Lester, J.C.: Enhancing 5th Graders' Science Content Knowledge and Self-Efficacy through Game-Based Learning. Computers & Education Journal. 59, 497–504 (2012).
8. Chaudhuri, S., Kumar, R., Howley, I., Rosé, C.P.: Engaging Collaborative Learners with Helping Agents. Proceedings of AIED. pp. 365–272 (2009).
9. Hayashi, Y.: On Pedagogical Effects of Learner-Support Agents. Proceedings of ITS. pp. 22–32 (2012).
10. Holland, J., Baghaei, N., Mathews, M., Mitrovic, A.: The Effects of Domain and Collaboration Feedback on Learning in a Collaborative Intelligent Tutoring System. Proceedings of AIED. pp. 469–471 (2011).
11. Esper, S., Foster, S.R., Griswold, W.G.: On the Nature of Fires and How to Spark Them When You're Not There. Proceedings of the SIGCSE Conference. pp. 305–310. ACM Press, New York, New York, USA (2013).
12. Stephenson, C., Wilson, C.: Reforming K-12 Computer Science Education… What Will Your Story Be? ACM Inroads. 3, 43–46 (2012).
13. The College Board: AP CS Principles: Learning Objectives and Evidence Statements, (2010).
14. Harvey, B., Mönig, J.: Bringing "No Ceiling" to Scratch: Can One Language Serve Kids and Computer Scientists? Constructionism. pp. 1–10 (2010).
15. Nagappan, N., Williams, L., Ferzli, M., Wiebe, E., Miller, C., Balik, S., Yang, K.: Improving the CS1 Experience with Pair Programming. Proceedings of the SIGCSE Conference. pp. 359–362 (2003).
16. Preston, D.: Pair Programming as a Model of Collaborative Learning: A Review of the Research. Journal of Computing Sciences in Colleges. 20, 39–45 (2005).
17. Sabourin, J., Rowe, J.P., Mott, B.W., Lester, J.C.: When Off-Task is On-Task: The Affective Role of Off-Task Behavior in Narrative-Centered Learning Environments. Proceedings of AIED. pp. 523–536 (2011).
18. Landis, J.R., Koch, G.G.: The Measurement of Observer Agreement for Categorical Data. Biometrics. 33, 159–174 (1977).
19. Williams, L.A., Kessler, R.R.: All I Really Need to Know about Pair Programming I Learned in Kindergarten. Communications of the ACM. 5, 108–114 (1999).
20. Arroyo, I., Cooper, D.G., Burleson, W., Woolf, B.P., Muldner, K., Christopherson, R.: Emotion Sensors Go to School. Proceedings of AIED. pp. 17–24 (2009).
21. Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B., Kafai, Y.: Scratch: Programming for All. Communications of the ACM. 52, 60–67 (2009).