Game2Learn: A study of games as tools for learning introductory programming concepts

Tiffany Barnes[1], Heather Richter[2],
Amanda Chaffin[1], Alex Godwin[1], Eve Powell[1]
1 - Department of Computer Science
2 - Department of Software and Information Systems
University of North Carolina at Charlotte
9201 University City Blvd., Charlotte NC
{tbarnes2, richter}@uncc.edu

Tiffany Ralph[3],
Paige Matthews[4], Hyun Jordan[5]
Department of Computer Science
3 - Colorado State University
4 - Wofford College, South Carolina
5 - Lynchburg College, Virginia

ABSTRACT

We introduce Game2Learn, an innovative way to teach introductory programming using games. Through an iterative development process, we are designing an educational framework that is more relevant and familiar to students' experiences, resulting in a game that will appeal to a broad audience and can improve student engagement, satisfaction, and skill transfer, particularly for women and other underrepresented minorities in computer science. In this paper, we discuss our first round of rapid prototyping, which explores different game and interface possibilities through two prototypes. Evaluations of these prototypes provide evidence that a game for learning programming can be fun, engaging and satisfying for students, and that a game can present programming concepts through a variety of formats and storylines.

1. INTRODUCTION

The goal of the Game2Learn Research Lab is to improve recruiting and retention in computer science through immersing computing instruction in game-based learning environments. We hypothesize that teaching introductory programming using a multiplayer online role-playing game can improve student engagement, satisfaction, and skill transfer, particularly for women and other underrepresented minorities in computing. We are employing a spiral software development cycle, alternating rapid prototypes with evaluation to investigate the effectiveness of a variety of game and interface possibilities.

We present the first round of our rapid prototyping, which resulted in two prototype role playing games designed to teach basic programming concepts. Results from this work will inform educators about the effects of game, plot, and interface design on the use of games to achieve the long-respected goals of making learning in STEM fields fun, engaging, concrete, motivating, collaborative and relevant.

In this paper, we describe background and related work, the process of game design and development for our two prototype games, the outcome of our usability study of these two games with computing students, and directions of future work.

2. BACKGROUND

Our nation's continued global competitiveness is widely believed to depend upon the U.S. maintaining its leadership in the development of new information technologies [3,22], but the number of students enrolling in IT programs is declining [33]. While enrollments of women and minorities in science and engineering programs are increasing [24], their enrollment in computer science is decreasing [33].

Although women are not well-represented in IT undergraduate programs, women make up forty percent of all gamers, and spend more of their game time than men playing with friends and family [19]. They make up just over half (50.4%) of online gamers [15], and twenty to thirty percent of massive multiplayer online role-playing games (MMORPGS) [21]. Games are increasingly being recognized to have built-in motivation and familiarity for most students [26,4,16], as well as incorporating expertise from a variety of computer science-related fields [20].

Games have successfully been used in computer science to motivate and inspire students in a variety of ways. Many courses replace some or all of their assignments with games, from introductory programming courses [26,28,31] up through senior level capstone courses [25] and software engineering [27]. A growing number of colleges and universities are adding game design and development courses to their curriculum to attract more students, in an effort to counter the trends of decreasing enrollments in computer science across the country [11]. Still other courses incorporate visual programming [8], computing for computation [17], computer graphics, multimedia, or virtual worlds into their courses to make the curriculum more concrete for students [13,29]. These courses are based on educational research that shows that grounding teaching in familiar, concrete, and relevant examples improves learning [7]. All of these research projects report success in terms of better grades and higher motivation and satisfaction of students. Gumhold and Weber [16] also break down their results by gender, and suggest that non-violent gaming assignments motivate women equally as well as men. However, diversity and gender have not been a primary focus of much of the research on games in teaching computer science, and therefore little is known about the specific effects of these interventions on recruitment and retention of diverse students. Researchers are still gathering evidence to determine whether these approaches instill in students skills and knowledge

that can be easily transferred to subsequent courses in the curriculum.

A large number of games on the market today are complex, multiplayer strategy and role playing games, such as World of Warcraft, Lineage II, and Guild Wars. MMORPGs require cognitive thinking, problem-solving, reading, mathematics, collaboration with other people, everything that goes with participating in a society effectively [30]. However, educators are only now incorporating the use of MMORPGS in the teaching of biology, physics, and economics [14, 32].

We propose to apply games in the computer science curriculum through playing an MMORPG. Instead of building games, defining the movements of characters in a virtual world [13, 18], or coding with specialized iconic programming languages [10], our students will play a collaborative multiplayer online role-playing game that requires programming to meet its challenges. In this approach, we can leverage the positive aspects of gaming (familiar, concrete, and engaging) and the research on games that appeal to a diverse audience, to counter the negative experiences that often occur in introductory computer science courses. The collaborative multiplayer aspects of the role-playing game create an environment that is inviting, safe, and socially stimulating for all students entering computer science.

We emphasize that in this approach, we need to design game goals to incorporate programming in a way that is directly transferable to subsequent courses. We believe it is critical to not only create an environment in computer science courses in which students are engaged but also to ensure that the skills learned remain relevant and transferable.

3. GAME PROTOTYPING

The development of a full-scale massively multiplayer online role playing game (MMORPG), such as one we envision for Game2Learn, requires thousands of man-hours, and in the games industry, the success of a game can often seem hit or miss. To balance the need for formative feedback to ensure success, and to accommodate smaller development teams, we have chosen to employ a highly iterative rapid prototyping development model, and to use existing game engine technologies to provide content including art, models, and sounds.

During the Summer of 2006, we built rapid prototypes to investigate a variety of in-game programming and learning techniques, as well as game story and setting, for their effectiveness and enjoyability for learning computer science. To begin the process of game development, we chose concepts from the IEEE and ACM joint curriculum for CS1 [ACM], including conditionals (if-then), iteration (for and while loops), and recursion. These topics were chosen for their universal relevance to CS1, and for increasing levels of difficulty, as is appropriate both for pedagogy and for game design.

After selecting concepts to teach, we brainstormed ideas for quests to teach each concept, and interfaces for students to learn programming. Some existing interfaces and methods that aid students in learning to program include Alice, where students code through list interaction[13], Karel J Robot, where students use Java with a minimum of syntax [5], DrJava, where students program interactively through a easy to use interface [2], Peridot, which uses visual icons to create programs [23], and jGrasp, which provides visualizations to support learning [12]. We decided that in these initial prototypes, we would explore alternative interfaces for learning programming concepts such as those above, that did not rely on typing. The interfaces we selected included fill in the blank, multiple choice, flow charts, dialog trees, and matching.

Several principles based on research in designing games and educational theory guided our current set of designs. Research suggests that males prefer epic struggles while females are more motivated by personal connections or improving someone's life or situation, implying that a theme that incorporates both of these ideas would be more gender-neutral [6]. The game must allow for experimentation and exploration, and have few ways for characters to completely fail or lose [9]. And the game story should be natural and programming skills should be seamlessly embedded in the gaming environment. While we will eventually create a game that contains both cooperative and individual tasks to better support learning, these initial prototypes focused solely on individual game play.

Through our brainstorming, we found a number of game possibilities we wished to explore. This resulted in two very different game prototypes. "Saving Princess Sera" is a two-dimensional exploratory game where the player learns of the kidnapping of the princess and determines to rescue her. "The Catacombs" is a linear three-dimensional game where the player is an apprentice wizard who must complete a sequence of tasks to rescue a family trapped in catacombs underground.

3.1 Saving Princess Sera

Saving Sera was developed using RPGMaker. The look and feel of Saving Sera is that of a children's game, with a simple, colorful 2-dimensional tile-based map accompanied by cheerful music, where players move and interact using only arrow keys and the space bar. To add a variety of programming interfaces to the game, our team built additional interface features, including typing and mouse controls using the RubyScript language.

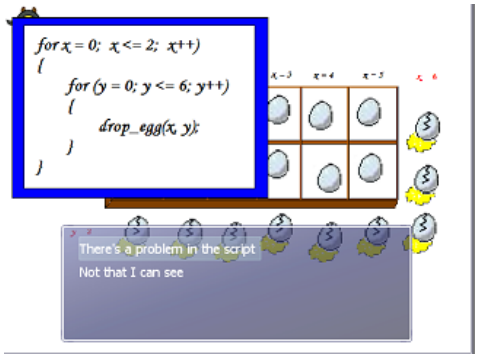

Figure 1: Egg Drop Quest in Saving Princess Sera

Saving Princess Sera is based on the notion of a village where a player may explore and perform tasks. The storyline is that Princess Sera is captured by a monster Gargamel. Arshes, a man from the village, hears this news and determines to rescue the princess. The player helps Arshes in his quest, first by completing programming tasks to earn enough money to buy passage on a boat to Gargamel's castle, and then by fighting alongside Sera to escape from the castle. A screen shot is shown in Figure 1.

The tasks each involve programming concepts: correctly reordering a while loop statement of a confused old fisherman's mind; correcting a nested for loop placing eggs in crates; and visually piece together a quicksort algorithm. When the player makes a mistake, Arshes must fight a script bug, which asks the users various computer science questions in order to fight the bug.

With this game, we investigated how users felt about a two-dimensional game, as well as a storyline where the user can explore and choose which people to talk to and tasks to perform.

3.2 The Catacombs

This three-dimensional game was developed using the BioWare Aurora toolset that was used to build NeverWinter Nights, a popular fantasy role-playing game based on Dungeons and Dragons. In this game, the storyline is that the player, an apprentice wizard, is asked by a mother to find her two young children who never returned from the catacombs. This is a linear game, where at each point there is only one possible task that the player may perform. We created three tasks, each progressively more complicated. The first involved unlocking the door to the catacombs involving two if statements; the second, building a bridge brick by brick with nested for loops; and the third, to solve a cryptogram using more nested for loops. In the 2nd and 3rd quests, incorrect answers result in decreasing player health. Each script developed for the quests is stored in the player's journal, which he or she can view while building the code, and refer to later in completing more difficult tasks in the game.

Figure 2: Casting a spell in The Catacombs

We created two versions of this game, to explore the same tasks with two different interfaces for creating the code. One, called Grimore, uses dialogue with a sarcastic spellbook named Grimore and multiple choice questions (a dialogue tree) to create the code and complete the task. In the second, called Konijn, the player chooses the correct scroll among many incorrect ones, or uses gemstones received in the game, which represent code snippets, to fill in the blanks of chunks of code. A screenshot of The Catacombs is shown in Figure 2.

4. EVALUATION

As part of our iterative design approach, we performed an exploratory evaluation of our prototypes. The goal of our prototypes was not to effectively teach any one concept, but instead to examine a number of game and interface options and overall feedback on the concept. Thus, we used students who had taken at least one introductory Computer Science course and were thus somewhat familiar with the programming concepts in the game. We gave each participant a demographic questionnaire and a pre-test of computer programming concepts. Each participant then spent approximately 20 minutes playing Saving Princess Sera and 20 minutes playing one version of The Catacombs. The participant was then given a post-test, and interviewed to gather his/her feedback about the game ideas. We asked students about their impressions of each game, and each quest within the game. We queried them about what they thought of using such a game as homework in a course, and if they would have liked such a method. We videotaped all gameplay and interviews, and created software logs of time-stamped interactions in The Catacombs.

We had 13 students participate in our evaluation. These students came from a variety of backgrounds: including 2 Asian females, 2 white females, 1 black male, 1 Hispanic male, and 7 white males. Most were ages 18-24, with 2 participants ages 25-30 and 1 aged 31-40. Nine participants are in computing related majors while the other four are studying electro-mechanical systems, psychology, communications, and art. Most participants had taken a class in either Java or C/C++. Participants included 5 sophomores, 4 juniors, 2 seniors, and 2 college grads.

The pre- and post-tests contained problems where students: determine the outcome of an if-then-else statement, convert a while loop into a for loop, detect an error in a simple maximum integer procedure, and give the output of a for loop containing a print statement. Despite their experience in programming courses, students performed surprisingly poorly on the pre- and post-tests, even several computer science majors. Majors averaged pre:5.9/post:5.8 and non-majors pre:2.25/post:2. We did not expect any real learning to occur with our prototypes given the short amount of time participants were playing. Instead, we used the post-test to see if any changes might have occurred, such as a re-familiarization with the programming concept. We essentially found no difference in pre- and post-test scores for students. Despite these poor test results, students were able to understand and complete most of the various programming quests, although with some mistakes being made. Some of these in-game mistakes, however, were clearly because of issues with our interfaces.

In Saving Princess Sera, few participants completed all of the game quests as they could spend a significant amount of time exploring the village, completing what quests they found and chose to perform. Most spent several (3-4) minutes initially wandering the village, and several performed non-programming tasks such as visiting the village store to purchase items. The two simpler tasks, the fisherman and egg quests, received the most positive feedback. These two quests were relatively straightforward and short, taking approximately 2 to 4 minutes to complete. Students found the print quest the most confusing, with most not using the provided in-game tutorial and not realizing 1) that they should type variable names in for the strings to display and 2) how to correct their mistakes. Participants who did play the more complex quicksort quest often thought this quest was too lengthy, playing 6 minutes, often ending by dying or not finishing. Successfully completing this final quest involved answering computer science questions to battle a bomb monster, however we did not make it very clear how to successfully defeat this bomb. While the results were tied to correctly answering questions, this was not evident to players.

More participants successfully completed all of the tasks in The Catacombs, particularly those playing the Grimore version. Players were better instructed as to what to do next. Six used

Grimore, seven the Konijn version. Players in both versions had difficulty in performing some tasks because they did not read or remember the directions to open a journal in order to view the code. Players in Konijn had even more difficulty, as the interface for choosing gem code snippets was not as useable as we would have liked it to be and participants often had to be given additional help to understand the interface. In general, subjects spent more time than expected on exploring the caverns, interacting with characters, and learning the game's controls. We received positive feedback on Grimore's sarcasm and often observed players laughing while playing both games.

Overall, subjects enthusiastically encouraged us to continue development. Students liked both Saving Princess Sera and The Catacombs overall, and found the games well balanced between easy and difficult and between play and quest time. Several students expressed that the games may still be challenging for very beginners. This is not surprising though as we were not targeting truly novice programmers with these sets of tasks. Although students felt that more instructions were needed for some quests, they "could see using the game as a re-enforcer for a class, something to do before a test." One subject in particular who had difficulty completing the quest in the allotted time responded to the question of whether or not he thought the game could be used to teach introductory programming, said, "Yeah! I mean, it would be awesome if like, after a lecture, the professor just said 'Alright, get to level 43 this weekend.' [He laughs.] I would have definitely wanted to be in that class. You guys should develop this into a game that's like that."

5. DISCUSSION

As our results indicate, the feedback about the games was overwhelmingly positive. Students often indicated that they enjoyed playing our prototypes, and that they could envision using these games as a course supplement. While we are still relying on student speculation, this feedback provides at least initial evidence that we can achieve our goals of providing an engaging learning environment.

While we did not see any strong patterns of gender differences, several students did comment that the Saving Sera prototype seemed more appealing to younger students or women, and indeed a few female students saying "I liked the girlie game," referring to Saving Sera. The game controls were easier to learn, and the graphics more simple, which may have contributed to this impression. However, this game also contained more battles than The Catacombs, which could have potentially turned women off to the game. We did not see any particular negative feelings, aside from being lost at first, about the Catacombs from any group of participants. Thus, while there may be differences in game preferences for men and women, we can not yet determine how much variety we will need to offer to have broad appeal.

Another issue that we uncovered is how students approach to the game, whether as a game or as homework, affected their comments. One student stated, "It's something other then mindless clicking. You actually have to think, something rarely seen in games today." This and a few other students seemed to think of our prototypes as mainly a game, and enjoyed playing such a game involving programming, but some were unsure if the game tasks would teach them enough. Students saw the potential for learning, though, stating, "Coding was easier, but still got harder as I went." Other students thought of the prototypes as potential homework, and thought a game would make homework more fun but questioned whether the game tasks would be serious enough assignments, especially since students could guess and eventually get correct answers on the game tasks. These comments reinforce our goals to provide in-game feedback that is closely related to the learning tasks. If this feedback is appropriate, whether students approach the tasks as a game or as homework, the feedback is relevant and motivating to them. We realized that in these prototypes, we did not provide very clear feedback that a player's health or battles were tied to correctly performing tasks. As a result, we are adding additional feedback mechanisms to both games to more visibly and strongly tie performance in the game with performance on the programming tasks. We are preparing a follow-up study to compare performance, length of time spent on tasks, and participant comments with this study to examine whether more appropriate feedback affects student motivation, behavior and impressions.

A final issue is the challenge of implementing a variety of interfaces for programming using existing game engines. Many of the interfaces we desired required modifications to the game engine's expected interaction, and several were not as useable as we would have liked, leading to some difficulties for users. Additionally, this initial evaluation did not help us determine which of our interface possibilities are best to pursue. Users did like interacting with Grimore, the sarcastic spellbook, so we will continue to explore the role of humor and dialogue in our games. Beyond that, we still need to continue to develop and investigate a variety of interfaces for programming in a game environment to determine the kinds of interaction we will support in the future.

6. CONCLUSION

We are investigating Game2Learn, an innovative way to teach introductory programming using games. We aim to design an educational framework that is more relevant and familiar to students' experiences, resulting in a game that will appeal to a broad audience and can improve student engagement, satisfaction, and skill transfer, particularly for women and other underrepresented minorities in computer science. In this paper, we have discussed our first round of rapid prototyping, which explored different game and interface possibilities through two prototypes. Evaluations of these prototypes provide evidence that a game for learning programming can be fun, engaging and satisfying for students, and that a game can present programming concepts through a variety of formats and storylines. Student comments brought out the potential importance of in-game feedback that is both related to the story and to the learning involved. We are currently examining this issue more closely with follow-up studies involving additional feedback mechanisms in our prototypes. Future Game2Learn designs will investigate in more detail issues of learning such as maintaining student motivation, and transfer of knowledge into traditional programming environments. Through our iterative design and evaluation process, we plan to implement a full-scale educational game that has potential to make significant contributions to educational research on technology and games and to study the effect of teaching practices on diversity in computer science.

7. ACKOWLEDGMENTS

This work was partially supported by the CRA Distributed Mentor Project and by NSF-**0552631** Computing REU Site at UNCC.

8. REFERENCES

[1] ACM/IEEE-CS Joint Curriculum Task Force. *Computing Curricula 2001*. Accessed Sep. 8, 2006. Online: http://acm.org/education/curric_vols/cc2001.pdf

[2] Allen, E., Cartwright, R., and Stoler, B. 2002. DrJava: a lightweight pedagogic environment for Java. *Proceedings of the ACM's 36th Technical Symposium on Computer Science Education (SIGCSE'02)*. (Cincinnati, Kentucky, Feb. 27 – Mar. 3, 2002). ACM Press, New York, NY, 137-141.

[3] Barnes, T., Berenson, S., & Vouk, M. On participation of women in information technology. In *Gender and Information Technology Encyclopedia*, The Pennsylvania State University Press, 2006.

[4] Becker, K. Teaching with games: The Minesweeper and Asteroids experience. *The Journal of Computing in Small Colleges Vol. 17*, No. 2, 2001, 22-32.

[5] Bergin, J. Karel universe drag & drop editor. In *ITICSE '06*. (Bologna, Italy, June 26 - 28, 2006). ACM Press, NY, 307.

[6] Bennett, D., Brummer, C., McDermott, M., & Green, L. Designing for diversity: Investigating electronic games as pathways for girls into information technology preofessions. In *Proceedings of the NSF ITWF & ITR/EWF PI Conference*, (Pittsburgh, PA, Oct. 24-26, 2004).

[7] Bransford, J., Brown, A., & Cocking, R. *How people learn: Brain, mind, experience and school*. National Academy Press, Washington, DC, 1999.

[8] Carlisle, M., Wilson, T., Humphries, J., & Hadfield, S. RAPTOR: a visual programming environment for teaching algorithmic problem solving. *SIGCSE'05*, (St. Louis, MO, Feb. 23-27, 2005). ACM Press, NY, 2005, 176-180.

[9] Case, S. Women in Gaming. Microsoft, Jan. 12, 2004. Online: http://www.microsoft.com/windowsxp/using/games/learnmore/womeningames.mspx

[10] Chen, S. & Morris, S. Iconic programming for flowcharts, Java, Turing, etc. In *Proceedings of the Integrating Technology in Computer Science Education Conference (ITiCSE 2005)* (Monte de Caparica, Portugal, June 27–29, 2005). New York, ACM Press, 2005.

[11] Coleman, R., Krembs, M., Labouseur, A., & Weir, J. Game design & programming concentration within the computer science curriculum. *SIGCSE 2005*: 545-550.

[12] Cross, J. jGRASP: an integrated development environment with visualizations for teaching Java in CS1, CS2, & beyond. *J. Comput. Small Coll. 21*, 4 (Apr. 2006), 118.

[13] Dann, W., Cooper, S., & Pausch, R. *Learning to program with Alice Beta Version*. Prentice Hall, 2005.

[14] Epstein, David. Joystick Nation. *Inside Higher Ed News,* Oct. 26, 2005. Online, accessed Nov. 15, 2005: http://insidehighered.com/news/2005/10/26/games

[15] Guernsey, L. 'Women Play Games Online in Larger Numbers Than Men,' *The New York Times*, January 4, 2001.

[16] Gumhold, M. & Weber, M. Motivating CS students with game programming. In *Proceedings of the 6th International Conference on New Educational Environments (ICNEE),* (Neuchatel, Switzerland, Sep. 27-30, 2004).

[17] Guzdial, M. A media computation course for non-majors. In *ITiCSE 2003*, New York, ACM Press, 2003, 104-108.

[18] Harvey, B. *Computer Science Logo Style: Symbolic Computing*. 2nd. Cambridge, MA, MIT Press, 1997.

[19] Jenkins III, H. Expanding and empowering the audience. The Education Arcade: Games Literacy Workshop, 2005. Online: http://www.educationarcade.org/

[20] Jones, R. Design and implementation of computer games: A capstone course for undergraduate computer science education, *SIGSCE 2000*, (Austin, TX), New York, ACM Press, 2000, 260-264.

[21] Laber, E. 'Men are from Quake, Women are from Ultima,' *The New York Times*, January 11, 2001.

[22] Malcom, S., Babco, E., Teich, A., Jesse, J.K., Campbell, L., & Bell, N. *Preparing Women and Minorities for the IT Workforce*. AAAS & CPST, 2005.

[23] Myers, B. A. 1990. Creating user interfaces using programming by example, visual programming, and constraints. *ACM Trans. Program. Lang. Syst. 12*, 2 (Apr. 1990), 143-177.

[24] National Science Foundation. Women, Minorities, and Persons With Disabilities in Science and Engineering:2000. Arlington, VA, (NSF 00-327): Appendix Table 5-15, p. 232.

[25] Parberry, I., Roden, T., & Kazemzadeh, M. Experience with an industry-driven capstone course on game programming: extended abstract. *SIGCSE 2005*: p91-95.

[26] Prensky, M. Digital Game-Based Learning, New York, McGraw-Hill, 2001.

[27] Rucker R. *Software engineering and Computer Games*, San Jose, Addison Wesley, 2002.

[28] Sindre, G., Line, S., Valvåg, O. Positive experiences with an open project assignment in an introductory programming course. In *Proc. 25th International Conference in Software Engineering (ICSE'03),* (Portland, OR, May 3-10, 2003).

[29] Stansfield, S. An introductory VR course for undergraduates incorporating foundation, experience and capstone. *SIGCSE 2005*: 197-200.

[30] Steinkuehler, C. Learning in massively multi-player online games. In *Proceedings of the Sixth International Conference on Learning Sciences*, (Mahweh, NJ, 2004), 521-528.

[31] Sweedyk, E., de Laet, M., Slattery, M., & Kuffner, J. Computer games and CS education: why and how. In *SIGCSE 2005*: 256-257.

[32] UNCG Economics 201: http://econ201.uncg.edu/

[33] Zweben, S. 2003-2004 Taulbee Survey: Record Ph.D. production on the horizon; undergraduate enrollments continue in decline. *Computing Research Association Taulbee Survey*, May 2005.