Game2Learn: Building CS1 Learning Games for Retention

Tiffany Barnes, Eve Powell, Amanda Chaffin, Alex Godwin
Department of Computer Science
University of North Carolina at Charlotte
9201 University City Blvd., Charlotte NC
tbarnes2@uncc.edu

Heather Richter
Department of Software and Information Systems
University of North Carolina at Charlotte
9201 University City Blvd., Charlotte NC
richter@uncc.edu

ABSTRACT
This paper presents Game2Learn, an innovative project designed to leverage games in retaining students in computer science (CS). In our two-pronged approach, students in integrative final-year capstone courses and summer research experiences develop games to teach computer science, which, in turn, will be used to improve introductory computing courses. Our successful model for summer undergraduate research and capstone projects engages students in solving the computing retention problem, allows them to quickly create games, and instructs students in user- and learner-centered design and research methods. Results show that this method of building games to teach engages students at multiple levels, inspiring newer students that one day their homework may all be games, and encouraging advanced students to continue on into graduate studies in computing.

Categories and Subject Descriptors
K.3.2 [**Computers and education**]: Computer and information science education. – *computer science education.*

General Terms
Design, Experimentation, Human Factors.

Keywords
Undergraduate research, capstone courses, games.

1. INTRODUCTION
The goal of the Game2Learn Research Lab is to improve recruiting and retention in computer science, particularly for women and other underrepresented minorities. Our unique approach leverages students' enthusiasm for games and social relevance to inspire advanced computer science students to create games for teaching introductory computer science. We present our structure for designing, developing, and testing Game2Learn games as capstone and research experiences, and the results from our initial usability tests. This structure may be an ideal application of games that can result in increased recruitment and retention at both ends of the undergraduate computing curriculum.

ITiCSE'07, June 23-27, 2007, Dundee, Scotland, United Kingdom.

1.1 Background
Although demand for computing jobs is rising, the number of students enrolling and graduating in computer science is declining, while the proportion of women and minorities is also decreasing [14]. Attrition rates in computing are as high as 30-40%, with most leaving after CS1 or CS2 [5]. Reasons for these high attrition rates include: poor knowledge about computing, poor math and problem-solving skills, poorly designed lab courses, lack of practice and feedback, and under-prepared instructors [5]. Today's students, from freshmen to seniors, desire relevant assignments that are engaging, challenging, and creative, that also prepare them for their careers. Game2Learn engages advanced students in helping to improve CS1 courses through incorporating up-to-date technologies, developing relevant assignments, and addressing individualized student needs. This learning occurs in the context of a capstone project, emphasizing a challenging balance between product and process [7,11,12].

Learning through building games is increasingly being used to motivate CS assignments, curricula, and undergraduate research [4,8,11,13], but little is known about its effects on retention. Well-known strategies for improving retention include engaging students in research [2,12] and building community [2]. Learning through playing games is becoming increasingly recognized for its inherent motivation [9,10,13]. However, instead of merely placing learning in a game, engagement must be woven in with the types of feedback most useful in learning [9].

Game2Learn research students learn about retention, diversity, education, game design, and research methods, and collaboratively apply this knowledge in creating a real solution to a real problem. This innovative combination, combined with support for the research and development process [as in 7,12], may contribute significantly to retention at both ends of the undergraduate computing curriculum.

2. GAME2LEARN PROJECT MODEL
The Game2Learn project began in Summer 2005 as a capstone project for computer science majors, with the goal of building educational games for teaching computer science that could make a difference for underrepresented minorities. From Summer 2005-Spring 2006 seven undergraduate and two graduate students, including three women and four blacks, worked with the first author on building the vision for Game2Learn. This exploratory research was successful in recruiting minorities, and one (female) project student applied to graduate school as a direct result of her participation in the project. However, the unstructured nature of the project was frustrating for some students, and initial outcomes were not well integrated or usable in teaching computer science. We believe this problem arose because we approached

Game2Learn from a game design rather than a teaching perspective, and left the design problem much more undefined.

Therefore, when the first two authors received support for six undergraduates in 10-week summer research experiences in Game2Learn, we designed a 10-week Game2Learn project model as shown in Table 1, to provide undergraduates the structure needed to achieve results. We chose to employ a highly iterative rapid prototyping development model, and to use existing game engine technologies to provide content including art, models, and sounds. These design choices, influenced by research methods in human-computer interaction, intelligent tutoring systems, and game development, allow students, from sophomores to graduating seniors, to participate much more equally and produce working games in a short amount of time.

Table 1: 10-week Game2Learn Project Model

Weeks	Goal(s)
All	Read relevant literature & keep research blog
1-2	Learn game engines by building small games
3	Select concepts & write sample target code
4	Brainstorm learning games for the concepts
4	Storyboard promising game ideas
5-7	Implement game prototypes
8-9	Playtest with potential users
10	Write & present results & literature reviews

Our hypotheses were: 1) Students participating in the summer program would be more likely to consider graduate study in computing, 2) Working as a group on this project would scaffold the experience for students of varying backgrounds, 3) Students would learn about computing research design and methods, 4) Students would be able to complete playable prototype games by the end of the summer, and 5) The results of the usability studies would inform the design of educational games for computing. These hypotheses were tested using student post-surveys, blogs, and research reports and presentations. In the remainder of this paper, we discuss the results of this model used in Summer 2006 in terms of both retention and resulting products, and its adaptation for Fall 2006, and we suggest ways to modify this model for semester-based capstone projects.

3. DEVELOPING GAMES

During the Summer of 2006, the first two authors mentored six students in undergraduate research. Half of these students had already taken the first author's course in game design. For their first 2 weeks, students learned to use engines including GameMaker, Unreal Tournament 2004, NeverWinter Nights, 3D GameStudio, and RPGMaker. Then, students chose to design games for concepts including conditional, iterative, and recursive structures from the ACM-IEEE computing curriculum [1] because of their universal relevance to CS1, and for increasing levels of difficulty, both for pedagogy and game design.

After selecting concepts, we brainstormed quest and interface ideas for learning to program in-game. Students were encouraged to explore alternative interfaces, and discouraged from simply building in-game IDEs. The interfaces selected included fill in the blank, multiple choice, flow charts, dialog trees, and matching. The group voted on the suggested ideas, and developed paper prototypes (i.e. storyboards) for the top ideas.

Students based their ideas on research in game design, diversity and retention, computer science education, feedback from local instructors, and pedagogy. The sequence of in-game activities was planned to scaffold learning: 1) players would observe a concept in action, 2) players would interactively change code to solve a game problem, getting to practice in a low-stress environment with help and feedback [9], and 3) players would eventually create their own code to solve in-game problems. Role-playing games were selected since they allow for experimentation and exploration, and have few ways for characters to completely fail [6]. They also allow many ways to seamlessly embed programming skills into a game story.

After brainstorming and storyboarding, students chose game engines to implement their projects and split into two groups to create two very different game prototypes, "Saving Princess Sera" and "The Catacombs", and designed a usability study to investigate which of the games were most motivating and easy to use. Due to the nature of the summer research, students had only about three weeks to prototype their games, two weeks to run the usability study, and one week to write their results. In the following sections, we describe the interactive learning games that students created, demonstrating that, when given a process to follow with clear goals for the resulting game, students can work quickly together to create and test learning games in a summer.

3.1 Saving Sera

Saving Sera is a two-dimensional exploratory game, implemented using RPGMaker, where the player learns of the kidnapping of the princess Sera and decides to rescue her. In the game, players with sufficient knowledge can create "machina" (which are really programs) to solve problems. This idea was similar to "god in the machine", a plot technique that introduces twists that resolve seemingly hopeless situations. In Saving Sera, the player fixes machina by unscrambling a while loop to track a fisherman's catch; debugging a nested for loop placing eggs in crates; and visually piecing together a flowchart for quicksort. When the player makes a mistake, he or she must fight a script bug by answering various computer science questions. The look and feel of Saving Sera is classic, with a simple, colorful map and cheerful music. To develop games in RPGMaker, students use a relatively simple drag-and-drop interface to create a tile-based map and dialog trees for characters. Since RPGMaker supports only arrow key and space bar interaction, students used RubyScript to modify the games with full keyboard and mouse controls. These modifications required students to: become familiar with a new scripting language, learn how the existing RPGMaker system worked, design new interaction techniques, and implement them.

In the Egg Drop Quest shown in Figure 1, the local Egg Factory owner has created a machina to drop eggs into crates, and offers the player a reward to fix it. The player answers multiple choice questions to debug the nested for loops, determining that the loops count too high. After interactively correcting the loops and watching the results run, the player's character explains to the owner why for loops seem to count one lower than the number needed: because the counting starts at zero!

In the Fishing Quest, students unscramble a do-while loop that helps a fisherman count the fish he's caught and decide when to stop. The machina is shown with comments, but the code is removed and placed in a list on the left of the screen. Players iteratively select the scrambled list items to place on each line, and "compile" their code, getting feedback on any errors. If the player decides to proceed without correcting errors, he or she fights a script bug by answering computing trivia questions.

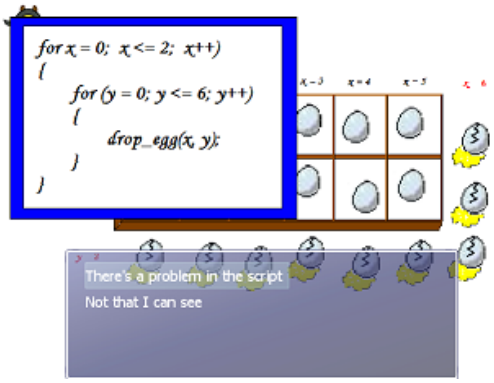

Figure 1: Egg Drop Quest in Saving Sera

In the Print Quest, students write code to make a small flyer using print statements and string variable names. They are given a list of statements such as **string1 = "Come to Ned's!"**, and fill-in-the-blank **print(_)** statements, and should type in the name of the string to print. An in-game tutorial was offered for this quest.

In the Quicksort Quest, students answer multiple choice questions about recursion, and construct a quicksort algorithm (to sort a box of keys and escape the castle) by dragging and dropping parts of the algorithm into a flow diagram.

3.2 The Catacombs

The Catacombs is a three dimensional fantasy role playing game developed using the BioWare Aurora toolset for NeverWinter Nights (NWN). In this game, the user is an apprentice wizard who must cast three increasingly complex spells (programs) to save two children who are trapped in the catacombs. The first spell uses two if statements to magically unlock a door, the second uses nested for loops to construct a bridge, and the third spell uses nested for loops to solve a cryptogram.

The three spells are not given to the player; instead, he or she must construct them in the game. Since one goal of the summer study was to determine which interfaces were most realistic, enjoyable, and effective for learning to program, students created two versions of this game, to explore the same tasks with two different interfaces: dialog trees and matching. In the Spellbook version, the player's personified sarcastic spellbook refuses to show any spells, instead constructing the spells interactively with the student as he or she correctly answers his questions. In the Gemstone version, the player chooses the correct spell among many incorrect ones, or fills in blanks in spells (programs) that have been partially worn off of ancient rune stones.

Developing NWN modifications involves detailed planning, scripting, and dialog tree construction. Designing non-traditional interactions into this RPG provided students with the challenge to examine a large code base and determine mappings between provided in-game interactions and the types of behaviors needed for learning to program. Students determined two innovations that allowed students to construct code and "fill in blanks". In the first innovation, students scripted the dialogs so that when players make choices in the game, the code being constructed interactively appears and is saved in their "journal", to visualize the program and save it for future reference. To model multiple choice behaviors, students scattered scrolls containing candidate spells in the level, and after collecting only the correct scroll and discarding all others, players can unlock the door to the catacombs. To model filling in blanks, students constructed rune stones that appear in the student's journal with numbered blanks missing. Players iteratively select code snippets (represented as gems) from their inventory to fill in each blank. Students envisioned that players would be able to write their own code and save it to their journals.

Figure 2. Casting a spell in The Catacombs.

4. PLAYTESTING

To prepare for this phase of the summer research, one student researched usability studies and worked with the professors to create the experimental design and pre- and post-tests. All students participated in recruiting, videotaping and transcribing gameplay and interviews.

Game2Learn games strive to balance play and learning time, to provide appropriate learning feedback, to maximize transfer of skills between in-game and real-world experiences, and to strongly tie in-game motivation to actual learning outcomes. As part of the iterative design approach to achieve these goals, we performed an exploratory evaluation of our prototypes with students who had already taken a CS1 course.

Participants took a demographic questionnaire and a pre-test of computing concepts, played about 20 minutes of each game, including Saving Sera and one version of The Catacombs, took a post-test, and was interviews about their experience. The interviews were used to gather what testers thought of using games like ours as homework, which games and/or quests they preferred, and how balanced the games were in play to quest time.

The 13 participants included 5 sophomores, 4 juniors, 2 seniors, and 2 college grads, mostly aged 18-24, with 4 females and 9 males, all of whom took a CS1 course. The 10-question pre- and post-tests asked students to predict outcomes for short conditional statements and for loops, convert while loops into for loops, and detect an error in computing a maximum integer. Although we did not expect significant learning gains over the study, scores were still surprisingly low, with the nine computing majors averaging pre:5.9/post:5.8 and non-majors pre:2.25/post:2. Despite poor test

results, students completed most quests, and were overwhelmingly enthusiastic about the idea of learning in games.

Some interesting lessons learned are:

- Students would love to play games for homework but are just as concerned as professors that the games should provide enough realistic and complex interaction for real learning.
- Clear instructions and game goals must be provided and accessible throughout the game.
- Learning goals must be clearly tied to in-game feedback that motivates the player (through, e.g. experience points, health), and penalizes guessing.
- The final game outcome must be very clear to provide a sense of accomplishment and completeness.
- Interactive in-game visualization of code is both engaging and effective for learning, (as in the Saving Sera Egg-drop Quest).
- The game should provide mechanisms and instructions for correcting mistakes before assessing quest results.
- Humor can be a motivation for in-game interaction.
- Implementing novel, usable interfaces for programming in-game can be challenging to build into existing game engines.
- The need for instructions and the tendency to view games purely as homework tools was much greater in players with little to no game experience.

We did not anticipate one effect: some students approached the games just as games, while others viewed the games as learning tools, and both sides questioned the seriousness of the games for homework. We hypothesized that better instructions and in-game feedback tied to learning outcomes would remove this concern and lead to more positive attitudes and in-game performance. After incorporating these changes into the Game2Learn games during Fall 2006, we ran another usability study [3] with 8 participants. Our results were positive, showing that while 54% of subjects in Study 1 felt games could be used as homework and 37% felt there was a good balance between quest and play time, 88% of the participants in Study 2 agreed that the games could be used for homework and 88% of the participants liked the play-quest balance or requested more quests. No Study 2 participants questioned the seriousness of the game. Although in-game performance (on time or errors) was not significantly affected, students completed a few tasks faster, and their scores tended to improve between the pre- and post- tests. We believe the in-game feedback provided better motivation, engagement, and reinforcement of learning outcomes.

5. EFFECTS OF MODEL ON STUDENTS

To test the effects of the summer research program, we conducted surveys and collected students' summer research blogs, final reports, and presentations. Survey questions included:

1. Did working on the project provide experience in how research is planned, conducted and reported?
2. Did working on the project enhance your interest to pursue research as part of your future career goals?
3. Describe your experience in working as a member of a research team? Was it valuable (comment in what respect) or would you have liked to work on it individually?

Four out of six students responded to the survey, indicating that the summer research project provided experience in how computing research is planned and conducted, but that a focus was placed on usability studies. All four respondents listed that the **project enhanced their interest in research and in graduate studies**, with one who was already strongly interested in research, and one who had never before considered doing research. Three local students continue to work on Game2Learn, while one student is conducting other game research at her own university.

All four respondents felt that **working as part of a team was very valuable**, but one of these students felt it was challenging since group members had varying goals, and expressed that it would be interesting to conduct individual research in the future. This response highlights that students who worked on the project before had a very different notion of how to approach the project than new students. Having veteran research members helped **scaffold experiences** for new students, but some **student expectations had to be adjusted** to the new model.

Final presentations demonstrated that, although students came from widely divergent backgrounds, all students could participate fully and were able to produce working games by the end of the summer. All students expressed some concern that their summer work was collaborative, while their final reports and presentations were individual, and they were unsure how to divide the literature review and report their efforts. In addition, the quality of literature reviews in final reports varied widely. To address these issues, we plan to develop and provide students with report-writing and literature review guidelines, and **frequent formal deliverables for reports and literature reviews.**

Playtest results have already contributed to our **understanding of feedback in educational games**. However, one respondent stated that results were not collected early enough in the summer to adequately analyze these results; a modified approach with **earlier playtesting** may provide students with practice in running a study and analyzing results to improve the games.

Overall, students indicated that **approaching the design from a learning objectives standpoint**, rather than developing a game and trying to add learning components, was much more **effective and satisfying**. These students have all spent a full year on the Game2Learn project, and commented that the **"results-driven"** approach, paired with dividing into **small groups** of 2-3 students was much easier to manage. Students also suggested that working on the project **full-time** aided in focus and productivity.

Students suggested that we should: 1) **shorten brainstorming time**, 2) **conduct formal research** on game and educational software interaction **early**, and 3) **lengthen development time**. These suggestions reflect student frustrations with designing many ideas that were never used, wanting more time to refine the ones that were, and finding better ideas later.

6. GAME2LEARN CAPSTONES

During Fall 2006, five new students began Game2Learn capstone projects. When informally surveyed, reasons our eight current students have given for joining the project are:

- We get to work in an established group on directed research to solve a real problem, and results will be used in real classes.
- We believe in the project and want to improve CS education.
- I am building my portfolio (games, educational software, etc.)
- Bragging rights!; I'm interested in MMORPGs.

- It was interesting to research what might be more effective in making learning games for women and minorities.

Both professors and veteran students were excited to use lessons from the summer to support more productive capstone projects. However, we had to modify the model to accommodate 1) completing summer research, 2) running a new study for an early fall deadline, and 3) balancing Game2Learn with other responsibilities. In their first few weeks, new Game2Learners participated in completing the summer studies, running Study 2, and demonstrating Game2Learn games for lab visitors. To reduce workload, less time was spent on brainstorming, and formal storyboarding was not required. New capstone students report:

- It was motivating to join in to an ongoing project, but hard to get started. I never expected that I would be presenting my work at the NC Undergraduate Research Symposium!
- It was slow getting started, since this project has a different way of doing everything [e.g. rapid prototyping, user studies].
- **Running the study** first helped me understand how to do research and what kinds of games to make.
- Sometimes I lose focus and work on the wrong things.

Small teams are working, but not as well as they did this summer, when groups could focus and work together for long periods. In the summer, extensive **brainstorming and storyboarding** seemed somewhat unnecessary, but students need these processes in the absence of constant collaboration and feedback.

7. CONCLUSION

We have refined our Game2Learn model into a set of guidelines for summer or capstone courses:

- Divide the groups into small teams of 2-3 students with at least one veteran member.
- New members start by playing past prototypes, modifying them, and making small games.
- Set background research, final report, and presentation guidelines early and require deliverables early and often.
- Place the Game2Learn work in context by having new participants conduct small studies with past prototypes.
- Conduct early brainstorming to cast a wide net for ideas, and require formal storyboards before any prototyping begins.
- Set prototype deliverables every two weeks and conduct early and frequent playtests and demonstrations.
- The professor should choose learning objectives and provide existing educational materials as guides for design.

These guidelines will help ensure continued success in recruiting and retaining advanced CS students and inspiring them to pursue research. Evaluations of Game2Learn prototypes provide evidence that a game for learning CS1 can be fun, engaging and satisfying. Participants highlighted the importance of in-game feedback that is related to both learning outcomes and gameplay.

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 and participation in computer science.

8. ACKOWLEDGMENTS

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

9. REFERENCES

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

[2] Aspray, W., & Bernat, A. Recruitment and retention of underrepresented minority graduate students in computer science: Report of a workshop, March 4-5, 2000. Washington, DC: Computing Research Association.

[3] Barnes, T., H. Richter. Game2Learn: Improving the engagement and motivation of CS1 students. Submitted to ACM SIGRAPH Symposium on Video Games 2007.

[4] Bayliss, J. & S. Strout. Games as a "flavor" of CS1. In SIGCSE2006. ACM Press, New York, NY, 500-504.

[5] Beaubouef, T. & J. Mason. Why the high attrition rate for computer science students: some thoughts and observations. SIGCSE Bull. 37, 2 (Jun. 2005), 103-106.

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

[7] Clear, T., Goldweber, M., Young, F. H., Leidig, P. M., and Scott, K. 2001. Resources for instructors of capstone courses in computing. SIGCSE Bull. 33, 4 (Dec. 2001), 93-113.

[8] Clua, E., B. Feijó, J. Schwartz, M. Graças, K. Perlin, R. Tori, T. Barnes. Games and Interactivity in Computer Science Education. Panel at SIGGRAPH, Boston, MA, August 2006.

[9] Garris, Ahlers, & Driskell. Games, motivation, and learning: a research and practice model. Simulation & Gaming, Vol. 33, No. 4, 2002, 441-467.

[10] Gee, J. P. What video games have to teach us about learning and literacy. Comput. Entertain. 1, 1 (Oct. 2003), 20.

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

[12] Polack-Wahl, J. & Anewalt, K. Undergraduate research: Learning strategies and undergraduate research. SIGCSE 2006: 209 – 213.

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

[14] Zweben, S. Computing Research Association Taulbee Survey, May 2005.