Assessing Game-Themed Programming Assignments for CS1/2 Courses*

Kelvin Sung
Computing and Software Systems
University of Washington, Bothell
ksung@u.washington.edu

Michael Panitz
Software Programming
Cascadia Community College
mpanitz@cascadia.ctc.edu

Rebecca Rosenberg
Teaching and Learning Center
University of Washington, Bothell
beckyr@u.washington.edu

Ruth Anderson
Computer Science and Engineering
University of Washington, Seattle
rea@cs.washington.edu

ABSTRACT

We have designed and implemented game-themed programming assignment modules targeted specifically for adoption in existing introductory programming classes. These assignments are *self-contained*, so that faculty members with no background in graphics/gaming can *selectively pick and choose* a subset to combine with their own assignments in existing classes. This paper begins with a survey of previous results, followed by a description of the game-themed assignment modules. The paper then focuses on our efforts in assessing the assignments: including details of the procedures and instruments for evaluating student achievement of learning outcomes and attitudes towards the assignments, as well as the extra effort required of faculty to adopt the assignments in existing clases.

Categories and Subject Descriptors

K.3.2 [**Computers and Education**]: Computer and Information Science Education – *computer science education*

General Terms

Design, Experimentation

Keywords

CS1/2, Games, Programming Assignments, Adaptation

1. INTRODUCTION

It has been demonstrated that teaching computer science (CS) concepts based on programming interactive graphical games motivates and engages students while accomplishing the desired student learning outcome (e.g., [30, 35]). Our own experience in integrating computer graphics and games programming [29] verified this observation. Inspired by the positive feedback from students and local employers, we started investigating possibilities of integrating games programming into other CS courses.

Because of the relatively straightforward technical topic areas and the abundant, favorable results (e.g., [6, 20]) we began our investigation with introductory programming courses (CS1/2). Our initial plan was to examine existing approaches, identify appropriate ones, and *gradually* integrate them into our courses. However, as discussed in the next section, most of the existing work is exploratory in nature and performed by faculty with expertise in gaming/graphics. Their results usually lack generality and do not facilitate gradual adoption by faculty that lack graphical/gaming expertise.

In 2006, we began our investigation into designing and developing CS1/2 game-themed programming assignments specifically targeted for *selective* adoption and limited-scope experimentation by faculty with no gaming/graphics experience [27]. Our work follows the observation that the modern generation of students demands multimedia experiences, i.e., real time *interaction* in a *visual* environment [14]. With existing results demonstrating that students' needs can be met (e.g., [7]) and learning outcomes can be consistent (e.g., [6]), our work focuses on the needs of the faculty. Our project consists of two phases. First, develop materials that are suitable for adoption. Second, adopt the materials in existing courses and then measure student acheivement of learning outcomes, student engagement and interest in the area, and the required faculty efforts.

The details of design and development of the game-themed assignments are discussed in [28]. In this paper, we first survey existing work done in combining CS1/2 courses with computer gaming. After that, we briefly describe the results from the first phase of the project and concentrate on the current efforts in assessing the effectiveness of the game-themed assignments.

Our assignments are simple "real-time interactive graphics programs". Strictly speaking, these programs do not qualify as "games" because they have unknown entertainment value. However, in our current implementation, the programs run on both PCs and the *XBOX 360* gaming console. As such, we use the term: "game-themed".

2. GAMES AND CS1/2 CLASSES

While there are many types of "games" that are suitable for teaching CS subjects (e.g., non-computer games [10])

*This work is supported in part by the National Science Foundation grant DUE-0442420 and Microsoft Research under the Computer Gaming Curriculum in Computer Science RFP, Award Number 15871.

GDCSE'08, February, 2008, Miami, Florida, USA.

our focus is interactive graphical computer games. As discussed in [29], recent work done in integrating gaming into CS classes can be classified into: *game development* (e.g., [24]) where students learn about building games; *game programming* (e.g., [30]) where students study games related algorithms; and *game client* (e.g., [29]) where students learn about CS concepts via games.

Integrating games into CS1/2 classes belongs to the *game client* category because after these classes students are expected to understand abstract programming concepts and not about building games. Based on the efforts from the faculty in preparing the programming assignments, existing work done in this area can be categorized into three broad approaches:

1. Little or no game programming assignments (e.g, [13, 15]). In these courses students learn by *playing* custom games but they do not actually program the games.

2. Per-assignment game development [11, 18, 7, 17, 26, 23, 31, 6]. All these classes developed games as part of individual programming assignments. In each case, isolated games are typically designed around technical topics area being studied.

3. Extensive game development. For example, faculty must develop and/or design programming assignments based on general games engines (e.g., [8]), dedicated games engine (e.g., [22]), specialized programming environments (e.g., [19]), custom object-oriented class hierarchies (e.g., [22]), specific curricula (e.g., [20]), or new programming languages (e.g., [9]).

Results from the last approach typically include large amounts of adoptable/adaptable courseware materials. However, using these materials usually requires significant investments, e.g., understanding a games engine, etc. Because of the considerable overhead, results from the third approach are not suitable for *selective* adoption.

In terms of suitability for *selective* adoption, we expect that the per-assignment game development approach would be best. For example, one could selectively replace non-game assignments in existing classes with the corresponding game assignments. However, because of the pioneering nature, many of the results are "anecdotal" without holistic considerations with respect to the CS1/2 curriculum. For example, [17] only involves turn-based strategic games, [26] only discuss puzzle games, and the discussion in [31] is based on one single game. The system described by [18, 23] are designed specifically for *object-first* approach. While results from [7] is based on an ascii character game, and the excellent ideas presented in [7, 6] are without implementations.

Our work is designed to be per-assignment game targeted to address the above issues. Our game-themed assignments are general CS1/2 assignments, require no existing knowledge in games or graphics from the faculty, and require only minimum changes to existing classes to be adopted.

3. THE ASSIGNMENT MODULES

We have developed 6 game-themed assignment modules based on the principles that the assignments should be: *neutral* with respect to gender and students' prior programming experience, *simple* enough that a minimum of resources, time, and effort are required from faculty to adopt the assignments, *modular* so that each assignment module is completely self-contained, and has sufficient supporting materials to facilitate independent adoption. Please refer to [28] for the details of the assignment design and development[1]. This section briefly describes the characteristics of the assignment modules.

3.1 Implementation Platform

We have chosen the C# programming language and the Microsoft XNA framework [3]. Our choice is governed by the fact that the Microsoft Games Studio Express, C#, and XNA combination is the only publicly available solution that provides seamless integration of the development environment, programming language, GUI API, and graphics API. Additionally and very importantly, XNA based programs can be compiled to run on the XBOX 360 consoles. This will provide an immediate draw to young students excited about being able to build games that can run on familiar entertainment devices in their first programming courses.

3.2 Choice of Topic Areas

The technical topics of our game-themed assignment modules are the same topics that are covered by our console-based assignments in our existing CS1/2 courses [2]. Choosing the topics this way provided the following advantages: (1) Our existing CS1/2 courses are successful and well-established. These facts justify the selected technical topic areas. (2) Assignments with identical technical topic areas imply they can be *interchanged*. This offers a perfect vehicle for the second phase of our project, where we can simply replace corresponding assignments and study the effects. (3) The console-assignments are included as part of the assignment modules. In this way, each assignment module addresses a well defined technical topic area and has two versions: a console version and a game-themed version. The console version of the assignment serves as an excellent and familiar reference for faculty members unfamiliar with graphics/games programming.

3.3 Contents of an Assignment Module

Each assignment module is designed to be self-contained, and consists of materials for both the faculty and the students.

For the faculty, each module includes a summary of the assignment, an game-themed version and a console-based version, a sample solution for both the console and game-themed versions, a sample grading rubric for each version, a list of prerequisites and expected learning outcomes for the students, a sample pre- and post- test, a list of frequently asked questions, and an *implementation tutorial*. The implementation tutorial is a step-by-step guide that explains the implementation of the game-themed (XNA-based) assignment, for faculty who are interested in learning more about how to create game-themed assignments on their own.

For the students, each module includes a description of the assignment, and skeleton starter projects. The game-themed starter project is a game-like application where all necessary graphics and user interactions are provided. Students work with the starter project to *fill in code*, using the relevant core CS concepts to complete each assignment,

[1]All materials described are available on-line at [1]

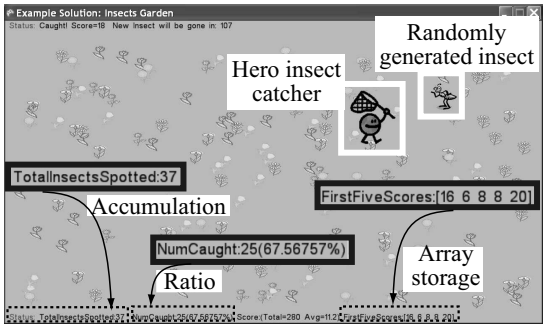

Figure 1: The Insect Garden Game.

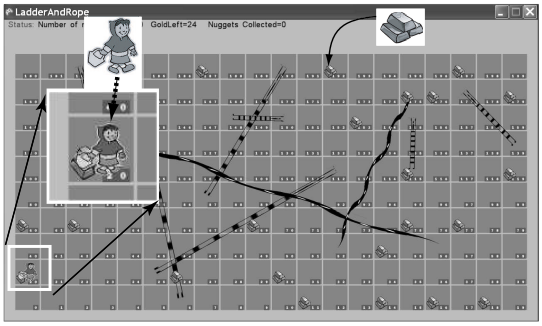

Figure 2: Ladder and Snake Game.

without having to know anything about XNA-specific implementation details.

3.4 Library Support

All of our game-themed assignments are implemented based on a simple $2D$ library. This library is designed to support faculty members interested in developing their own game-themed assignments. To maintain simplicity, the library consists of a single superclass for all assignments, a handful of drawing functions, and an instance of an input object for handling input. Our design agrees with results from previous approaches, where only simple primitives are required for CS1/2 courses [25].

3.5 Examples

As mentioned, the game-themed assignments are simple "interactive graphics applications". Here we use two of the assignments as examples to provide a *taste* of what the programs are like.

Example 1: This is one of the earlier assignments in CS1 on random number generators and operators. Figure 1 shows the game-themed version, where the user controls the *hero insect catcher* to net randomly generated insects. Provided with all the graphics and interaction functionality, students must support random number generation and maintain proper accumulated results, success ratio, etc. The console version of this assignment is on monte carlo integration where students must approximate the area of a circle based on randomly generated sample positions. In both cases, the finished solution consists of tens of lines of C# code in a single source code file, in addition to the code that was provided.

Example 2: This is an assignment on array of references to objects. Figure 2 shows that the game-themed version is a variation of the *Snakes and Ladders* game. Similar to the classic game, the heroine advances across the game board (a 1D array) based on die rolls and skips forward/backward upon encountering ladders/snakes. In our case, the goal of the game is to collect all the *gold bars* scattered on the game board. In addition, the user can request the creation of a ladder/snake in an empty game cell (creation of objects). As in all of our game-themed assignments, in this case the student starter project includes proper support for a completed game board with all necessary interaction functionality. The console version of this assignment is a periodic table where the user can query/insert element objects into a 1D array. In both cases, students only need to concentrate on managing object allocations/initializations and insertions into the array. Both finished solutions are similar, involving about 150 lines of C# code in two separate classes, in addition to the code that was provided.

4. ASSESSING THE ASSIGNMENTS

The game-themed assignments are designed for students to practice and learn fundamental concepts in programming. It is important to independently evaluate the academic contents of the materials. Additionally, it is important to verify the technical equivalence between the game-themed and console assignments in each module. Professor Ruth Anderson is our external reviewer. She is an experienced instructor who has taught CS1/2 courses many times at multiple institutions and in a variety of programming languages; has won multiple teaching awards (e.g., [5]); and is active in CS education research (e.g., [4, 16]). In addition to these excellent credentials, Professor Anderson is perfectly suited for evaluating our materials because she has never taught a graphics or gaming course and has limited experience with GUI programming. Before this project, Professor Anderson did not know anyone on the project team. During the project, we avoided in-person or verbal communications to maintain impartiality, and to simulate investigations by curious faculty. The assessment of the assignments was conducted across our project web-site [1] where newly released materials were downloaded, examined, and tested by Professor Anderson. Feedback was provided to us via a custom assignment evaluation form.

The assignment evaluation form is designed to collect both formative feedback and quantitative scores [12]. Each assignment module is assessed in two areas.

1. *Quality of the assignment*, assesses the merit of, and the technical equivalence between, the console and game-themed assignments, as well as the merit of the supporting materials (e.g., pre/post test, etc.).

2. *Potential for adoption*, is designed to assess the factors independent of the quality of assignments that may prevent adoption (e.g., programming language used).

Based on the review feedback, the assignments have been well received, oversall. We believe because the assignments were based on our existing CS1/2 assignments, technical merits were never an issue. Professor Anderson agrees with us that the assignments are appropriate for *typical* CS1/2 courses. Assignments with low *quality of assignment* scores were revised and re-assessed. This process continued until

the formative comments are positive and the numeric scores are above 4 (out of 5).

As expected, all of the low scores were caused by game-themed assignments. Typically, the reasons are in the following categories.

1. **Differences in difficulties**: our initial attempts at game-themed assignments often result in highly difficult, complex, or intimidating programs. Based on the feedback, we have adjusted the assignments accordingly.

2. **Inappropriate implementation of concepts**: designing assignments around *negative* results from game play is a bad idea. For example, a linked list structure might be used for tracking when the player has been unsuccessful at a given task. In this case, in order to test the linked list, students must purposely be unsuccessful at playing the game. This can take the fun out of the assignment.

3. **Deficiency in support**: the development team often overlook important details. For example, in the beginning, a specialized XBox 360 game controller (hardware) was the only way to play a game. Based on the feedback, this problem was remedied with a keyboard-based software object simulation.

Because Java is the language of choice at Professor Anderson's institution, we have received consistently lower scores for *potential for adoption*. We are fully aware that the language issue must be addressed for wide adoption of our results. Now that we have experience building these assignments, and understand the important attributes of the library, we are investigating possibilities in porting our results to other environments.

5. ASSESSING STUDENT LEARNING

The results presented in the previous section are initial verifications for the academic contents of the game-themed assignments. With these results, we can proceed with the second phase of our assessment: studying the effects of using these assignments in existing classes. We will study the adoption of game-themed assignment modules from three different aspects: student learning outcomes; student perception; and faculty efforts. This section describes our procedure, discusses our goals and the instruments designed in assessing each of these aspects.

5.1 Student Learning Outcomes

The feedback from the independent faculty evaluator indicated academic appropriateness, however, it does not mean that the game-themed assignments will be effective in the classroom. We must verify that we can *replace* individual console-based assignment with the corresponding game-themed assignment without negative impact.

Of the 6 assignment modules we have developed, 4 are targeted for CS1, and the other 2 are meant for CS2. We will replace 2 of the 4 console-based assignments in each of our existing CS1/2 classes [2]. In alignment with our teaching schedule, the game-themed assignments will be integrated and studied over three academic quarters:

Fall 2007	**Winter 2008**	**Spring 2008**
CS1-Games	CS1-Console	CS1-Games
CS2-Console		CS2-Games

The *CS2-Console* during Fall 2007, and *CS1-Console* during Winter 2008 are offerings of our existing CS1/2 courses without modifications. The results from these courses will serve as our control groups. We will replace 2 of the 4 existing console assignments with the corresponding game-themed assignments in each of the *CS1-Games* and *CS2-Games* courses from Fall 2007 and Spring 2008 . In this way, with the two offerings of the *CS1-Games* courses, and one offering of the *CS2-Games* course, we will have the opportunity to examine all 6 of the game-themed assignments. Notice that the *games-courses* consist of 2 existing console based and 2 new game-themed assignments. In this way we can verify that a faculty member can indeed select and replace *some* of the existing assignments.

In all cases, pre- and post- tests will be assigned as written homework assignments before and after each console and game-themed programming assignments. Together with students' mid-term and final exam results, we hope to verify that student learning outcomes are similar in all cases.

5.2 Student Perspective

Similar to much of the existing work in integrating computer gaming into CS curriculum [35, 21], one of the major motivations of our work is to further engage students via *interesting* and/or *relevant* assignments. With the recent decline in enrollments [33] and the decrease in student diversity [32], many people have pointed to the potentials of using highly interactive media applications as ways to connect with the new generation of students (e.g., [14]). In our case, we do not believe one or two game-like graphical assignments in CS1/2 courses will cause fundamental changes. However, we are interested in assessing students' interests and general attitudes towards the CS discipline. In addition, we are highly concerned with potential undesirable effects of computer gaming (e.g., [34]) and would like to assess students' perception of our game-themed assignments.

We have designed pre- and post- course survey forms to assess students' background, attitudes towards the course, and their interests in continuing with the CS major. Additionally, we have designed a per-assignment feedback form where students are asked to:

- verify the clarity of the assignment;
- predict their grades for the assignment;
- self-reflect on learning from the assignment; and
- evaluate the assignment in terms of interests, difficulty, satisfaction, and usefulness of time spent.

These survey forms will be filled out by the students in all of the above courses for all of the assignments, including the console-based assignments. By comparing results to the control group, we hope to identify any changes in students' level of interest and/or enthusiasm after the game-themed assignments.

5.3 Faculty Perspective

As discussed, *selective* adoption of the assignments by faculty with minimal experience in games and graphics programming is an important goal of our work. Towards this end, we must understand faculty's perception when using these assignments.

We have designed a custom survey form for the adopting faculty. On this form, the faculty is asked to reflect upon:

- their own agenda in adopting the assignments (e.g., reasons and goals, etc.);
- the *extra* efforts involved in using the assignment (e.g., extra time in grading or answering questions);
- independent from test results, their perception of student learning and enthusiasms;
- their perception of the assignments in terms of difficulty and appropriateness; and
- general difficulty in integrating the assignment with the rest of their courseware materials.

This information will provide guideline on how we can improve the supporting materials associated with each assignment module.

Discussion

At this point, we have completed the first phase of our project and verified that the academic content of the game-themed assignment modules are sound. We have designed all the instruments and procedures for evaluating the three aspects of using these assignments in existing classes: student learning, student perception, and faculty perception. We are currently using these assignments and are in the process of collecting assessment results. It is hoped/expected that the assessment results from the second phase of the project will affirm the effectiveness of the game-theme assignments.

6. REFERENCES

[1] XNA-based assignment home page: http://faculty.washington.edu/ksung/research/xna_games.

[2] BIT142/143: Intermediate programming and data structure, 2007. Cascadia Community College.

[3] XNA game studio, 2007. Microsoft Inc, *http://msdn2.microsoft.com/en-us/directx/Aa937794.aspx*.

[4] R. Anderson, R. Anderson, K. M. Davis, N. Linnell, C. Prince, and V. Razmov. Supporting active learning and example based instruction with classroom technology. *SIGCSE Bull.*, 39(1):69–73, 2007.

[5] R. E. Anderson. Acm faculty award, 2004. voted best teacher by students, Dept of CS, Univ. of Virginia.

[6] J. D. Bayliss and S. Strout. Games as a "flavor" of cs1. In *SIGCSE '06*, pages 500–504, 2006. ACM Press.

[7] K. Becker. Teaching with games: the minesweeper and asteroids experience. *J. Comput. Small Coll.*, 17(2):23–33, 2001.

[8] K. Bierre, P. Ventura, A. Phelps, and C. Egert. Motivating oop by blowing things up: an exercise in cooperation and competition in an introductory java programming course. In *SIGCSE '06*, pages 354–358, 2006. ACM Press.

[9] W. Dann, S. Cooper, and R. Pausch. *Learning to Program with Alice*. Prentice Hall, Upper Saddle River, NJ, 2006.

[10] P. Drake. *Data Structures and Algorithms in Java*. Prentice Hall, Upper Saddle River, NJ, 2006.

[11] N. Faltin. Designing courseware on algorithms for active learning with virtual board games. In *ITiCSE '99*, pages 135–138, 1999. ACM Press.

[12] J. Frechtling, L. Sharp, and Ed. *User-Friendly Handbook for Mixed Method Evaluations*. Directorate for Education and Human Resources, Division of Research, Evaluation and Communication, National Science Foundation, 1997.

[13] R. Giguette. Pre-games: games designed to introduce cs1 and cs2 programming assignments. In *SIGCSE '03*, pages 288–292, 2003. ACM Press.

[14] M. Guzdial and E. Soloway. Teaching the nintendo generation to program. *Commun. ACM*, 45(4):17–21, 2002.

[15] P. Haden. The incredible rainbow spitting chicken: teaching traditional programming skills through games programming. In *ACE '06*, pages 81–89, Darlinghurst, Australia, Australia, 2006. Australian Computer Society, Inc.

[16] T. B. Horton, R. E. Anderson, and C. W. Milner. Work in progress-reexamining closed laboratories in computer science. *FIE 2004*, October 2004. Conference Proceedings (Conference CD).

[17] T. Huang. Strategy game programming projects. In *CCSC-NE '01*, pages 205–213, , USA, 2001. Consortium for Computing Sciences in Colleges.

[18] R. Jiménez-Peris, S. Khuri, and n.-M. Marta Pati˙Adding breadth to cs1 and cs2 courses through visual and interactive programming projects. In *SIGCSE '99*, pages 252–256, 1999. ACM Press.

[19] M. Külling and P. Henriksen. Game programming in introductory courses with direct state manipulation. In *ITiCSE '05*, pages 59–63, 2005. ACM Press.

[20] S. Leutenegger and J. Edgington. A games first approach to teaching introductory programming. In *SIGCSE '07*, pages 115–118, 2007. ACM Press.

[21] M. Lewis, S. Leutenegger, M. Panitz, K. Sung, and S. Wallace. Introductory programming courses and computer games. In *Panel Proposal, Submitted to SIGCSE '08, Sep 2007*.

[22] M. C. Lewis and B. Massingill. Graphical game development in cs2: a flexible infrastructure for a semester long project. In *SIGCSE '06*, pages 505–509, 2006. ACM Press.

[23] T. Lorenzen and W. Heilman. Cs1 and cs2: write computer games in java! *SIGCSE Bull.*, 34(4):99–100, 2002.

[24] I. Parberry, T. Roden, and M. B. Kazemzadeh. Experience with an industry-driven capstone course on game programming: extended abstract. In *SIGCSE '05*, pages 91–95, 2005. ACM Press.

[25] E. S. Roberts. A c-based graphics library for cs1. In *SIGCSE '95*, pages 163–167, 1995. ACM Press.

[26] J. M. Ross. Guiding students through programming puzzles: value and examples of java game assignments. *SIGCSE Bull.*, 34(4):94–98, 2002.

[27] K. Sung. Xna based games-themed programming assignments for cs1/2. *Microsoft Research, Computer Gaming Curriculum in Computer Science, Award Number: 15871*, 2006-2008.

[28] K. Sung, M. Panitz, S. Wallace, R. Anderson, and J. Nordlinger. Game-themed programming assignments: The faculty perspective. In *Submitted to SIGCSE '08, Sep 2007*.

[29] K. Sung, P. Shirley, and B. R. Rosenberg. Experiencing aspects of games programming in an introductory computer graphics class. In *SIGCSE '07*, pages 249–253, 2007. ACM Press.

[30] E. Sweedyk, M. deLaet, M. C. Slattery, and J. Kuffner. Computer games and cs education: why and how. In *SIGCSE '05*, pages 256–257, 2005. ACM Press.

[31] D. W. Valentine. Playing around in the cs curriculum: reversi as a teaching tool. *J. Comput. Small Coll.*, 20(5):214–222, 2005.

[32] J. Vegso. Cra taulbee trends: Female students & faculty, 2005. *http://www.cra.org/info/taulbee/women.html*.

[33] J. Vegso. Drop in cs bachelor's degree production. *Computing Research News*, 18(2), March 2006.

[34] H. M. Walker. Do computer games have a role in the computing classroom? *SIGCSE Bull.*, 35(4):18–20, 2003.

[35] U. Wolz, T. Barnes, I. Parberry, and M. Wick. Digital gaming as a vehicle for learning. In *SIGCSE '06*, pages 394–395, 2006. ACM Press.