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 selective, incremental adoption in existing introductory programming classes. These assignments are self-contained, so that faculty members with no background in graphics or 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, and based on this survey, the paper summarizes the important considerations when designing materials for selective adoption. The paper then describes our design and implementation efforts followed by presentation of examples of the game-theme assignment modules. After which, the paper discusses our efforts in assessing the assignments: including details of the procedures and instruments for evaluating student learning outcomes, students' attitudes towards the assignments, and the extra effort required of faculty in adopting the assignments in existing classes. Our result is a road map that guides faculty members in experimenting with game-themed programming assignments by incrementally adopting and/or customizing suitable materials for their own classes.

* 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.

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 outcomes (e.g., [1, 2]). Our own experience in integrating computer graphics and games programming [3] verified this observation. Inspired by the positive feedback from students and local employers, we started investigating possibilities of integrating games programming in other CS courses.

Because of the relatively straightforward technical topic areas and the abundant favorable results (e.g., [4, 5]) we began our investigation with introductory programming courses (CS1/2). Our initial plan was to examine existing results, identify appropriate ones, and gradually integrate these results into our courses. However, as discussed in the next section, most of the existing works are exploratory in nature, and performed by faculty with expertise in gaming/graphics. The results of these works usually lack generality and do not facilitate gradual adoption by typical faculty.

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 [6]. Our work follows the observation that the modern generation of students demands multimedia experiences, i.e., real time interaction in a visual environment [7]. With existing results demonstrating that students' needs can be met (e.g., [8]) and learning outcomes can be consistently achieved (e.g., [4, 9]), 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 study the impact on student learning outcomes, on student engagement and interest, and on the effort required of faculty. The details of design and development of the game-themed assignments were first described in [10] while our initial assessment efforts were discussed in [11]. This paper consolidates these discussions for a holistic presentation of our project.

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, since the programs run on both PCs and the XBox 360 gaming console we use the term: "game-themed".

2. Games and CS Classes

There are many types of "games" that are suitable for teaching CS subjects including many non-interactive games (e.g., [12]) or games that are based on dedicated devices (e.g., Lego robots [13]). Our focus is interactive graphical computer games. As discussed in [3], when examining recent efforts in integrating gaming into CS classes, we observe three general categories.

1. Games development classes. These are entire curricula (e.g., [14-18]), individual classes (e.g., [19-25]), or capstone projects (e.g., [26, 27]) designed specifically to develop new games as an end product. When evaluated against the curriculum framework proposed by the IDGA education committee [28], we see that these classes cover all the major core topic areas. Students in these classes must be concerned with all aspects of a real game production including entertainment

value, visual quality, audio effects, physics simulations, and real-time performance.

2. Games programming classes. These are classes (e.g., [1, Kuffner's CMU course]) designed specifically to study technical aspects and issues involved in building games. For example, topics covered may include event loops, path planning algorithms, terrain representation, etc. These classes typically do not require building an end product and the topics covered are general and typically can be applied to different domains. These classes concentrate on covering the game programming topic area in the IDGA curriculum framework.
3. Games development client. These are existing CS classes that creatively integrate games into their existing curriculum. Typically, games are used as programming assignments (e.g., [1, 3, 4, 10, 29-34]) or to teach abstract concepts (e.g. [8, 31, 35-37]), or as an example application area to teach the concepts involved in an entire topic area (e.g., [38, 39]). These are traditional CS classes that exist independent from games programming. These classes are actually clients of games development where they use games development as a vehicle to deliver specific abstract concepts. After these classes, students are expected to understand the abstract concepts, and not about games development.

Games and CS1/2 Classes

Integrating games into CS1/2 classes belongs to the games development client category because after these classes students are expected to understand abstract programming concepts, rather than concepts specific to 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 (e.g., [31, 40]). In these courses students learn by playing custom games but they do not actually program the games.
2. Per-assignment game development [4, 8, 9, 30, 41-45]. All these classes developed games as part of individual programming assignments. In each case, isolated games are 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., [46]), dedicated games engines (e.g., [33]), specialized programming environments (e.g., [47]), custom object-oriented class hierarchies (e.g., [33]), specific curricula (e.g., [5]), or new programming languages (e.g., [36]).

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, significant reworking of an instructor's existing curriculum, etc. Because of the considerable overhead, results from the third approach are typically not suitable for selective adoption.

In terms of suitability for selective adoption, we expect that the per-assignment game development approach would be the best. For example, one would selectively replace non-game assignments in existing classes by the corresponding game assignments. However, because of the pioneering nature, many of the results are "anecdotal" and do not discuss the impact of such assignments on the CS1/2 curriculum holistically. For example, [42] only involves turn-based strategic games, [43] only discuss puzzle games,

and the discussion in [45] is based on one single game. The system described by [41, 44] are designed specifically for object-first approach. While results from [8] is based on an ASCII character game, and the excellent ideas presented in [8, 4] are without implementations.

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

3. Design Considerations

Based on the above survey of prior work, we articulate the following considerations for designing game-themed programming assignments targeted for selective adoption.

- Institutional Oversight: Departmental committee consensus is often required for significant changes to the core courses. Because of the potential impact, it can be especially challenging to arrive at consensus for modifications to introductory level courses like CS1/2. However, limited scope experimentation by individual faculty is usually acceptable. Our assignment modules are limited in scope and self-contained, to facilitate selective experimentation by individuals. In this way, faculty members can gain experience (and collect results) to assist the committee's decision making process.
- Faculty Experience: Many faculty members did not grow up playing computer games and most are not familiar with graphics programming. To facilitate adoption, game-themed assignment modules should be skeleton applications with complete graphics and user interaction functionality. In this way, faculty and students only need to concentrate on the missing core CS concepts. In addition, tools and support should be provided for the interested faculty to develop their own game-themed assignments. It is important that the provided tools define the proper abstractions to hide the details of real-time graphics programming.
- Gender and Expertise Neutrality: As with any powerful tool, inappropriate use of games can backfire and result in further alienation of under-represented groups [48]. It is important that the materials built are gender and expertise neutral. For example, it is important to avoid violence (shooting), and unnecessary competitions [49-51]. The materials should discourage the addition of superfluous "eye-candy" graphical enhancements, or user interaction programming by students with extensive prior programming experience. Doing so will help to avoid intimidating other, less experienced students.
- Infrastructure Support: Free and simple are the keywords here. Given the financial reality of most schools, all materials must be freely available; the associated institutional infrastructure requirements must be modest and straightforward.
- Conceptual Integrity: Since our work focuses on the needs of the faculty, the discussion and analysis have been centered on the perspective of the adopting faculty. It is important to remind ourselves that ultimately, the goal of this work is to facilitate students' learning about core CS concepts. Any dilution, even in favor of acquiescing to some students' desire and motivation to become game developers, would do the students a disservice.

4. Implementation Guidelines

The above design considerations dictate the following guidelines for our implementation efforts.

1. Neutrality, with respect to both gender and programming experience. The importance of this cannot be overstated, as we cannot afford to alienate under-represented groups.
2. Simplicity, with respect to resource requirements (as opposed to being technically easier for the students). Simpler is better, unless the instructional integrity of the faculty member's existing course materials is an issue.
3. Modularity, meaning that each assignment module should be completely self-contained, and should include sufficient supporting materials that faculty can consider the adoption of each assignment in isolation.

Based on these principles, the next section details our implementation efforts. All materials mentioned in next section are available on-line at [52].

5. The Assignment Modules

The first decision is the technical topics for the assignments. Becker [8] and Bayliss [4] are excellent references for ideas that relate fundamental programming concepts to existing commercial video games. In our case, we must be concerned with any unnecessary complexity; we need simple interactive graphical applications that assist students in implementing the relevant programming concepts. At the same time, we must ensure that our assignments are interchangeable with those of typical CS1/2 courses.

5.1 Choice of Topic Areas

We have chosen the topics for our assignment modules using a "reverse adoption" strategy; we have adopted the technical topics for the game-themed assignments based on the topics in our existing console-assignments in our CS1/2 courses [53]. There are several advantages to this strategy.

1. Our existing CS1/2 courses are well-established with many batches of successful alumni in advanced CS courses and in industry. This success justifies 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 is an excellent and familiar reference for faculty members unfamiliar with games programming.

We have implemented 6 assignment modules. We are using these to replace the corresponding assignments in our existing CS1/2 courses. The 6 modules cover topic areas that include integer division and the modulus operator, random number generation, single-dimensional arrays of object references, 2D arrays, class hierarchy/inheritance, and linked lists and queues. The next section will describe the assessment procedure that

ensures that the console and game-themed versions of the assignment are technically equivalent.

5.2 Implementation Platform

There has been work done in integrating the tools required for building interactive graphical computer games into introductory programming courses, including event handling (e.g., [54]), graphical user interfaces (GUI) (e.g., [55]) and graphical application programming interfaces (API) (e.g., [56]). In our case, to maintain simplicity, we are interested in hiding these aspects of games programming. We need a platform that transparently integrates all of the above tools so that faculty and students do not need to be aware of their existence.

We have chosen the C# programming language and the Microsoft XNA framework [57]. Our choice is governed primarily by the fact that C#, XNA, and Microsoft's Games Studio Express combination is the only publicly and freely available solution that provides seamless integration of the development environment, programming language, GUI API, and graphics API[1] .

A very important benefit of our choice is the fact that XNA based programs can run on both the PC and the XBOX 360 platforms. With our secondary goal of developing an introductory games programming course (e.g., [40]), this benefit is of special importance.

5.3 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 page describing the assignment, including prerequisite kmowledge that students must already possess, and a list of expected student learning outcomes
- a sample pre-and post-test
- a sample solution for both the console and game-themed versions
- a sample grading rubric for each version
- a list of frequently asked questions
- an implementation tutorial.

The implementation tutorial is a step-by-step guide that explains the implementation of the game-themed (XNA-based) assignment, and is intended to help interested faculty better understand how to create a games-themed assignment, using the XnaAssignmentBase library that was developed to support this project.

For the students, each module includes:

- a description of the assignment, and
- a skeleton starter project for both the console and game-themed version.

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 the relevant core CS concepts to complete each assignment, without having to know anything about XNA-specific implementation details. In this way, the assignments can maintain

[1] At the time when the project started, our main reservation with a Java based approach was the prospect of working with the separate Java3D library.

gender neutrality and programming experience neutrality, because students are not expected (or encouraged) to develop any games from scratch.

5.4 XnaAssignmentBase: The Supporting Library

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.

```
// Label A: XnaAssignmentBase Subclass responsibilities:
   public Constructor(...)                        // construct and define drawing dimension
   protected override void InitializeWorld()  // called once at init time
   protected override void UpdateWorld()    // called every 25 msec
   protected override void DrawWorld()      // called every 25 msec

// Label B: Output drawing support
   void EchoToBottomStatus(String msg)
   void EchoToTopStatus(String msg)
   void DrawCircle(Vector2 at, float radius,...String imageFile)
   void DrawRectangle(Vector2 center,...dimension... String imageFile)

// Label C: Input support
   XnaAssignmentBase.GamePad. ....          // poll object for state of input device
```

Listing 1: XnaAssignmentBase: The Supporting Library.

Listing 1 shows all the functions in our library. With this library, the only requirement for faculty to develop real-time graphical applications is an understanding of the ***UpdateWorld*** function (which updates game state) and the ***DrawWorld*** function (which draws all graphical primitives). Under Label **B** are the output drawing functions. Notice that, besides text output, the only supported graphical primitives are squares and circles. Our design agrees with results from previous approaches, which indicated that only simple primitives are required for CS1/2 courses [58]. The GamePad object under Label **C** contains instances of relevant button and trigger objects that reflect the state of the input device. Associated with this library is a step-by-step tutorial explaining each of the above functions within a simple interactive graphical application.

The simplicity of the library is noteworthy, as this simplicity enables a wide range of uses. There is no graphical class hierarchy or any utility functions (e.g., collision detection). Our design decision is based on experiences from building graphics libraries [3]. We have learned that clean and efficient utility functions enforce requirements on graphical objects and the complexity can increase rapidly resulting in elaborate class hierarchies. As discussed in recent results (e.g., [59, 55]), such hierarchies may not be well suited for supporting different approaches to teaching CS1/2 classes [60]. Moreover, many games (e.g., Reversi [45]) do not require any additional functionality from our library. Our goal is to facilitate experimentation by novice faculty members, and so simplicity is the key.

5.5 Example Assignments

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.

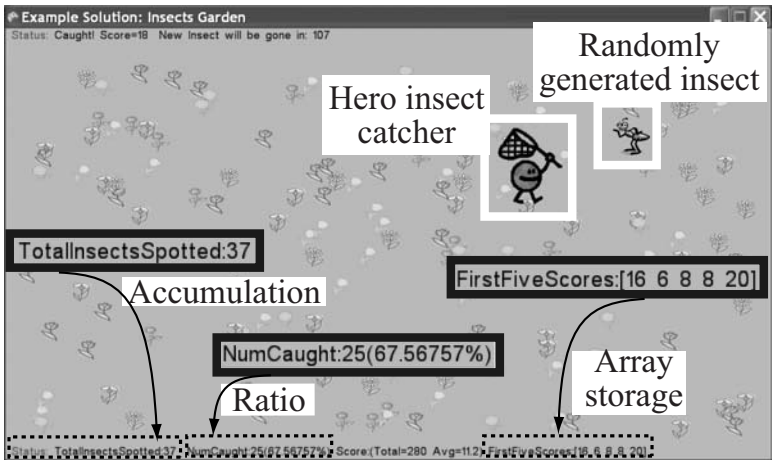

Figure 1: The Insect Garden Game.

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 solution consists of tens of lines of C# code in a single source code file.

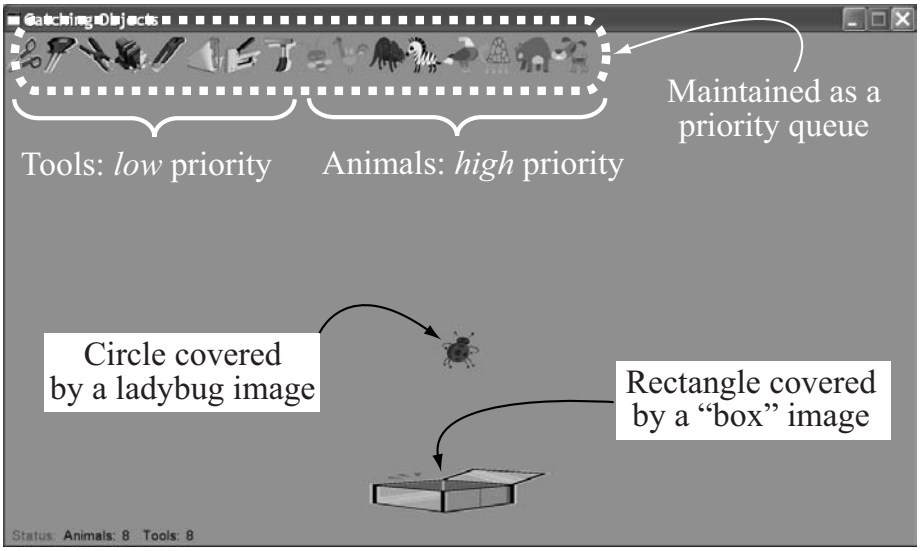

Figure 2: The Catch Objects Game.

Example 2: This is an assignment on linked lists based queues. Figure 2 depicts the game-themed version, in which the player can insert either high priority objects (animals) or low priority objects (tools) into the overall queue located at the top of the window. The game continuously dequeues and drops the oldest high-priority object (or the oldest low-priority object, if no high priority objects are available). The user moves the box to catch the dropping object. Students must implement the overall queue class, typically by maintaining items in one of two different (linked-list based) queue objects, based on each item's priority. The game interfaces with the overall queue class, and draws all the items in single row, so it appears that the objects are all in a single queue, regardless of student implementation. The console version of this assignment is a text-based "help desk" application. The user can enter high or low priority requests to be enqueued to the front/back queue respectively. Similar to the game-themed version, the retrieval of requests is a simple dequeue operation on the queues. The skeleton starter projects for both versions contain all necessary I/O functionality: graphics/GUI for game-themed, and character I/O for console. In both cases, students only need to implement the linked list queue and the priority queue. The solutions are almost identical involving about 200 lines of C# code.

6. 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.

6.1 Independent Reviewer

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., [61]); and is active in CS education research (e.g., [62, 63]). 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.

6.2 Procedure

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 [52] 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 [64]. Each assignment module is assessed in two areas.

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

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

6.3 Results

Based on the review feedback, the assignments have been well received, oversall. We believe because the assignments were reversely adopted based on existing CS1/2 classes, 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 continues 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 use of concept: 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 overlooks important details. For example, in the beginning, specialized hardware controller was the only way to control 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.

7. 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: study the effects of using these assignments in existing classes. We 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.

7.1 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 [53]. 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 CS2Games 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 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.

7.2 Student Perspective

As described, similar to many of the existing work in integrating computer gaming into CS curriculum [2, 11], 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 [65] and the decrease in student diversity [66], many people have pointed to the potential of using highly interactive media applications as ways to connect with the new generation of students (e.g., [7]). In our case, we do not believe one or two game-like graphical assignments in CS1/2 courses will cause fundamental changes in student engagement. However, we are interested in assessing students' interest and general attitude towards the CS discipline. In addition, we are highly concerned with potential undesirable effects of computer gaming (e.g., [48]) 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 interest in continuing with the CS major. Additionally, we have designed a per-assignment feedback from 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 difficulty, satisfaction, level of interest evoked, and usefulness of time spent.

These survey forms will be filled out by 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 interests and/or enthusiasm after the game-themed assignments.

7.3 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.

References

[1] Elizabeth Sweedyk, Marianne deLaet, Michael C. Slattery, and James Kuffner. Computer games and cs education: why and how. In SIGCSE '05: Proceedings of the 36th SIGCSE technical symposium on Computer science education, pages 256-257, New York, NY, USA, 2005. ACM Press.

[2] Ursula Wolz, Tiffany Barnes, Ian Parberry, and Michael Wick. Digital gaming as a vehicle for learning. In SIGCSE '06: Proceedings of the 37th SIGCSE technical symposium on Computer science education, pages 394-395, New York, NY, USA, 2006. ACM Press.

[3] Kelvin Sung, Peter Shirley, and Becky Reed Rosenberg. Experiencing aspects of games programming in an introductory computer graphics class. In SIGCSE '07: Proceedings of the 38th SIGCSE technical symposium on Computer science education, pages 249-253, New York, NY, USA, 2007. ACM Press.

[4] Jessica D. Bayliss and Sean Strout. Games as a "flavor" of cs1. In SIGCSE '06: Proceedings of the 37th SIGCSE technical symposium on Computer science education, pages 500-504, New York, NY, USA, 2006. ACM Press.

[5] Scott Leutenegger and Jeffrey Edgington. A games first approach to teaching introductory programming. In SIGCSE '07: Proceedings of the 38th SIGCSE technical symposium on Computer science education, pages 115-118, New York, NY, USA, 2007. ACM Press.
[6] Kelvin Sung. Xna based games-themed programming assignments for cs1/2. Microsoft Research, Computer Gaming Curriculum in Computer Science, Award Number: 15871, 2006-2008.
[7] Mark Guzdial and Elliot Soloway. Teaching the nintendo generation to program. Commun. ACM, 45(4):17-21, 2002.
[8] Katrin Becker. Teaching with games: the minesweeper and asteroids experience. J. Comput. Small Coll., 17(2):23-33, 2001.
[9] Jessica D. Bayliss. The effects of games in cs1-3. Journal of Game Development, 2(2), 2007.
[10] Kelvin Sung, Michael Panitz, Scott Wallace, Ruth Anderson, and John Nordlinger. Game-themed programming assignments: The faculty perspective. To Appear in SIGCSE '08, March 2008.
[11] Kelvin Sung, Michael Panitz, Becky Reed Rosenberg, and Ruth Anderson. Assessing game-themed programming assignments for cs1/2 courses. To Appear in GDCSE'08, Feb 2008.
[12] Peter Drake. Data Structures and Algorithms in Java. Prentice Hall, Upper Saddle River, NJ, 2006.
[13] Myles McNally, Michael Goldweber, Barry Fagin, and Frank Klassner. Do lego mindstorms robots have a future in cs education? In SIGCSE '06: Proceedings of the 37th SIGCSE technical symposium on Computer science education, pages 61-62, New York, NY, USA, 2006. ACM Press.
[14] Ron Coleman, Mary Krembs, Alan Labouseur, and Jim Weir. Game design & programming concentration within the computer science curriculum. In SIGCSE '05: Proceedings of the 36th SIGCSE technical symposium on Computer science education, pages 545-550, New York, NY, USA, 2005. ACM Press.
[15] Michael Zyda. Guest editor's introduction: Educating the next generation of game developers. Computer, 39(6):30-34, 2006.
[16] Janet Murray, Ian Bogost, Michael Mateas, and Michael Nitsche. Game design education: Integrating computation and culture. Computer, 39(6):43-51, 2006.
[17] Tracy Fullerton. Play-centric games education. Computer, 39(6):36-42, 2006.
[18] Lawrence Argent, Bill Depper, Rafael Fajardo, Sarah Gjertson, Scott T. Leutenegger, Mario A. Lopez, and Jeff Rutenbeck. Building a game development program. Computer, 39(6):52-60, 2006.
[19] Ian Parberry, Max B. Kazemzadeh, and Timothy Roden. The art and science of game programming. In SIGCSE '06: Proceedings of the 37th SIGCSE technical symposium on Computer science education, pages 510-514, New York, NY, USA, 2006. ACM Press.
[20] Bruce Maxim. Game design and implementation 1 and 2, 2006. Microsoft Academic Alliance Repository Newsgroup, Object ID: 6227, http://www.msdnaacr.net/curriculum/pfv.aspx?ID=6227.
[21] B. Burd, J. Goulden, B. Ladd, M. Rogers, and K. Stewart. Computer games in the classroom, or, how to get perfect attendance, even at 8 am. In SIGCSE '07: Proceedings of the 38th SIGCSE technical symposium on Computer science education, New York, NY, USA, 2007. ACM.
[22] Scott A. Wallace and Andrew Nierman. Using the java instructional game engine in the classroom. J. Comput. Small Coll., 23(2):47-48, 2007.
[23] Bill Clark, Jerry Rosenberg, Terrel Smith, Stu Steiner, Scott Wallace, and Genevieve Orr. Game development courses in the computer science curriculum. J. Comput. Small Coll., 23(2):65-66, 2007.
[24] Alexander Repenning and Andri Loannidou. Broadening participation through scalable game design. In SIGCSE '08: Proceedings of the 39th SIGCSE technical symposium on Computer science education, New York, NY, USA, 2008. ACM.
[25] Daniel Frost. Ucigame, a java library for games. In SIGCSE '08: Proceedings of the 39th SIGCSE technical symposium on Computer science education, New York, NY, USA, 2008. ACM.
[26] Randolph M. Jones. Design and implementation of computer games: a capstone course for undergraduate computer science education. In SIGCSE '00: Proceedings of the thirty-first

SIGCSE technical symposium on Computer science education, pages 260-264, New York, NY, USA, 2000. ACM Press.

[27] Ian Parberry, Timothy Roden, and Max B. Kazemzadeh. Experience with an industry-driven capstone course on game programming: extended abstract. In SIGCSE '05: Proceedings of the 36th SIGCSE technical symposium on Computer science education, pages 91-95, New York, NY, USA, 2005. ACM Press.

[28] IDGA. IGDA curriculum framework report version 2.3 beta, February 2003. International Game Developer's Association, http://www.igda.org/academia.

[29] Joel C. Adams. Chance-it: an object-oriented capstone project for cs-1. In SIGCSE '98: Proceedings of the twenty-ninth SIGCSE technical symposium on Computer science education, pages 10-14, New York, NY, USA, 1998. ACM Press.

[30] Nils Faltin. Designing courseware on algorithms for active learning with virtual board games. In ITiCSE '99: Proceedings of the 4th annual SIGCSE/SIGCUE ITiCSE conference on Innovation and technology in computer science education, pages 135-138, New York, NY, USA, 1999. ACM Press.

[31] Ray Giguette. Pre-games: games designed to introduce cs1 and cs2 programming assignments. In SIGCSE '03: Proceedings of the 34th SIGCSE technical symposium on Computer science education, pages 288-292, New York, NY, USA, 2003. ACM Press.

[32] Kelvin Sung and Peter Shirley. Algorithm analysis for returning adult students. The Journal of Computing Sciences in Colleges, 20(2):62-72, December 2004. Proceedings of the Sixth Annual CCSCNW Conference.

[33] Mark C. Lewis and Berna Massingill. Graphical game development in cs2: a flexible infrastructure for a semester long project. In SIGCSE '06: Proceedings of the 37th SIGCSE technical symposium on Computer science education, pages 505-509, New York, NY, USA, 2006. ACM Press.

[34] Woei-Kae Chen and Yu Chin Cheng. Teaching object-oriented programming laboratory with computer game programming. IEEE Transactions on Education, 50(3):197-203, August 2007.

[35] Stuart Hansen. The game of set®: an ideal example for introducing polymorphism and design patterns. In SIGCSE '04: Proceedings of the 35th SIGCSE technical symposium on Computer science education, pages 110-114, New York, NY, USA, 2004. ACM Press.

[36] Wanda Dann, Steve Cooper, and Randy Pausch. Learning to Program with Alice. Prentice Hall, Upper Saddle River, NJ, 2006.

[37] Paul V. Gestwicki. Computer games as motivation for design patterns. In SIGCSE '07: Proceedings of the 38th SIGCSE technical symposium on Computer science education, pages 233-237, New York, NY, USA, 2007. ACM.

[38] Flvio Soares Corra da Silva. Software engineering for computer games, 2006. University of Sao Paulo (USP/SP), Microsoft Academic Alliance Repository Newsgroup, Object ID: 6211, http://www.msdnaacr.net/curriculum/pfv.aspx?ID=6211.

[39] Flvio Soares Corra da Silva. Artificial intelligence for computer games, 2006. University of Sao Paulo (USP/SP), Microsoft Academic Alliance Repository Newsgroup, Object ID: 6210, http://www.msdnaacr.net/curriculum/pfv.aspx?ID=6210.

[40] Patricia Haden. The incredible rainbow spitting chicken: teaching traditional programming skills through games programming. In ACE '06: Proceedings of the 8th Austalian conference on Computing education, pages 81-89, Darlinghurst, Australia, Australia, 2006. Australian Computer Society, Inc.

[41] Ricardo Jimenez-Peris, Sami Khuri, and no-Martinez Marta Pati. Adding breadth to cs1 and cs2 courses through visual and interactive programming projects. In SIGCSE '99: The proceedings of the thirtieth SIGCSE technical symposium on Computer science education, pages 252-256, New York, NY, USA, 1999. ACM Press.

[42] Timothy Huang. Strategy game programming projects. In CCSC '01: Proceedings of the sixth annual CCSC northeastern conference on The journal of computing in small colleges, pages 205-213, , USA, 2001. Consortium for Computing Sciences in Colleges.

[43] John Minor Ross. Guiding students through programming puzzles: value and examples of java game assignments. SIGCSE Bull., 34(4):94-98, 2002.

[44] Torben Lorenzen and Ward Heilman. Cs1 and cs2: write computer games in java! SIGCSE Bull., 34(4):99-100, 2002.

[45] David W. Valentine. Playing around in the cs curriculum: reversi as a teaching tool. J. Comput. Small Coll., 20(5):214-222, 2005.
[46] Kevin Bierre, Phil Ventura, Andrew Phelps, and Christopher Egert. Motivating oop by blowing things up: an exercise in cooperation and competition in an introductory java programming course. In SIGCSE '06: Proceedings of the 37th SIGCSE technical symposium on Computer science education, pages 354-358, New York, NY, USA, 2006. ACM Press.
[47] Michael Kulling and Poul Henriksen. Game programming in introductory courses with direct state manipulation. In ITiCSE '05: Proceedings of the 10th annual SIGCSE conference on Innovation and technology in computer science education, pages 59-63, New York, NY, USA, 2005. ACM Press.
[48] Henry M. Walker. Do computer games have a role in the computing classroom? SIGCSE Bull., 35(4):18-20, 2003.
[49] In Justine Cassell and Henry Jenkins, editors, From Barbie to Mortal Kombat: Gender and Computer Games. MIT Press, Cambridge, Massachusetts, 1998.
[50] Denise G urer and Tracy Camp. An acm-w literature review on women in computing. SIGCSE Bull., 34(2):121-127, 2002.
[51] Marc J. Natale. The effect of a male-oriented computer gaming culture on careers in the computer industry. SIGCAS Comput. Soc., 32(2):24-31, 2002.
[52] XNA-based assignment home page http://depts.washington.edu/cmmr/research/xna games.
[53] BIT142/143: Intermediate programming and data structure, 2007. Cascadia Community College.
[54] Henrik Baerbak Christensen and Michael E. Caspersen. Frameworks in cs1: a different way of introducing event-driven programming. In ITiCSE '02: Proceedings of the 7th annual conference on Innovation and technology in computer science education, pages 75-79, New York, NY, USA, 2002. ACM Press.
[55] Viera K. Proulx, Jeff Raab, and Richard Rasala. Objects from the beginning -with guis. In ITiCSE '02: Proceedings of the 7th annual conference on Innovation and technology in computer science education, pages 65-69, New York, NY, USA, 2002. ACM Press.
[56] Sarah Matzko and Timothy A. Davis. Teaching cs1 with graphics and c. In ITICSE '06: Proceedings of the 11th annual SIGCSE conference on Innovation and technology in computer science education, pages 168-172, New York, NY, USA, 2006. ACM Press.
[57] XNA game studio, 2007. Microsoft Inc, http://msdn2.microsoft.com/en-us/directx/Aa937794.aspx.
[58] Eric S. Roberts. A c-based graphics library for cs1. In SIGCSE '95: Proceedings of the twenty-sixth SIGCSE technical symposium on Computer science education, pages 163-167, New York, NY, USA, 1995. ACM Press.
[59] Kim B. Bruce, Andrea Danyluk, and Thomas Murtagh. A library to support a graphics-based object-first approach to cs 1. In SIGCSE '01: Proceedings of the thirty-second SIGCSE technical symposium on Computer Science Education, pages 6-10, New York, NY, USA, 2001. ACM Press.
[60] Kim B. Bruce. Controversy on how to teach cs 1: a discussion on the sigcse-members mailing list. SIGCSE Bull., 37(2):111-117, 2005.
[61] Ruth E. Anderson. Acm faculty award, 2004. voted best teacher by students, Dept of CS, Univ. of Virginia.
[62] T. B. Horton, R. E. Anderson, and C. W. Milner. Work in progress-reexamining closed laboratories in computer science. ASEE/IEEE Frontiers in Education 34th Annual Conference, October 2004. Conference Proceedings (Conference CD).
[63] Richard Anderson, Ruth Anderson, K. M. Davis, Natalie Linnell, Craig Prince, and Valentin Razmov. Supporting active learning and example based instruction with classroom technology. SIGCSE Bull., 39(1):69-73, 2007.
[64] Joy Frechtling, Laure 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.
[65] Jay Vegso. Drop in cs bachelor's degree production. Computing Research News, 18(2), March 2006.
[66] Jay Vegso. Cra taulbee trends: Female students & faculty, 2005. http://www.cra.org/info/taulbee/women.html.