Game-Themed Programming Assignments For Faculty: A Case Study*

Cinnamon Hillyard
U. of Washington, Bothell
chillyard@uwb.edu

Robin Angotti
U. of Washington, Bothell
rrider@uwb.edu

Michael Panitz
Cascadia Comm. College
mpanitz@cascadia.edu

Kelvin Sung
U. of Washington, Bothell
ksung@uwb.edu

John Nordlinger
Microsoft Research
johnnord@microsoft.com

David Goldstein
U. of Washington, Bothell
dgoldstein@uwb.edu

ABSTRACT

Despite the proven success of using computer video games as a context for teaching introductory programming (CS1/2) courses, barriers including the lack of adoptable materials, required background expertise (in graphics/games), and institutional acceptance still prevent interested faculty members from experimenting with this approach. The *G*ame-*T*hemed programming *A*ssignment (GTA) modules are designed specifically for these faculty members such that they can selectively pick and choose a subset to experiment with and gradually adopt the materials in their own classes. The design and academic merits of the GTA modules have been verified and presented previously [24]. This paper begins by describing results from GTA workshops for CS1/2 faculty and goes on to detail the results of our year-long project in adopting the GTA modules in classes. In this case, we have demonstrated that introductory programming concepts can be examined, practiced, and learned based on GTA modules when neither the faculty nor the students involved have backgrounds in graphics or games. More importantly, our results showed that it is straightforward to *blend* the GTA modules into existing classes with minimum alterations. The GTA modules are excellent catalysts enabling faculty to begin exploring teaching with game-themed materials.

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, Assessment, Games, Assignments, Adaptation

*This work is supported in part by Microsoft External Research under the Computer Gaming Curriculum in Computer Science RFP, Award Numbers 15871 and 16531.

SIGCSE'10, March 10–13, 2010, Milwaukee, Wisconsin, USA.

1. INTRODUCTION

Proper integration of interactive graphical games into introductory programming (CS1/2) courses can motivate and engage students and build excitement and enthusiasm for the Computer Science (CS) discipline as well as accomplish desired student learning outcomes (e.g., [19, 3]). Most of the existing research in this area is based on pioneering exploratory projects by faculty members with expertise in computer graphics and games [20]. With few exceptions, these projects are "*student-centric*" where the main goals of study are student engagement and various learning outcomes. Adaptability and generality of the resulting materials are not primary concerns. For general faculty members teaching CS1/2 courses, few of which have computer graphics or games backgrounds, it can be challenging to take advantage of these results.

One way to support faculty members to explore teaching CS1/2 courses in the context of interactive graphical games is by building and freely disseminating game-themed educational modules that are *limited in curriculum scope* and *self-contained*. Such modules would allow faculty members to experiment and gain experience with game-themed materials in a well-defined sub-topic area as part of their normal classes and to demonstrate results to assist the decision-making process of institutional oversight committees [24]. The CS1/2 *G*ame-*T*hemed *A*ssignment (GTA) modules are *limited in curriculum scope* because each module is a programming assignment designed around a core computer science (CS) concept (e.g., binary search tree). In addition, each GTA module is completely independent and is a *self-contained* unit with extensive supporting materials including a detailed implementation tutorial.[1] In this way, faculty members can choose to experiment with any subset of the GTA modules to combine with their own assignments in their classes.

In the first of our two-phase project we have successfully designed, developed, and verified the academic merits of the GTA modules [24, 23].[2] This paper describes the en-

[1]Other materials include: a description of the assignment, prerequisite knowledge, and expected student learning outcomes; sample pre- and post-tests; a technically equivalent traditional console-based assignment; sample solutions for both the console-based and game-themed versions; sample grading rubrics; frequently asked questions; and sample student starter projects.

[2]All GTA modules are freely available at: *http://depts.washington.edu/cmmr/Research/XNA_Games/*.

couraging outcomes from our on-going GTA workshops for CS1/2 faculty and then presents the results from our year-long adoption of the GTA modules. It is important to note that there is nothing magical about teaching with games. As highlighted by Bayliss [4], faculty buy-in and experience are some of the most important factors in realizing the student engagement potentials of game-themed teaching approach. GTA modules are designed as catalysts for development of faculty expertise in the area. For this reason, in our study we do not anticipate evidence of overwhelming student engagement. Instead, the goals of our study are to verify that GTA modules can be adopted with minimal effort by a faculty member with no background in computer graphics or games, and with minimal disturbance to an existing course. We want to verify that the GTA modules "***do no harm***" while faculty members experiment with and develop experience in game-themed contexts.

The GTA modules 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 platform, we use the term, "game-themed."

2. BACKGROUND

Existing work on presenting CS1/2 concepts in the context of computer games can be broadly categorized into three approaches [23]. First, little or no games programming (e.g, [11]) where students learn by *playing* custom games. Second, per-assignment games development (e.g., [3, 24, 17]) where individual programming assignments are computer games designed around technical topics being studied. Third, extensive games development where faculty and students work with custom games engines (e.g., [5, 16]), specialized programming language (e.g., [8]), environments (e.g., [13]), or specific curricula (e.g., [14]), etc.

As discussed by Levy and Ben-Ari [15] and Ni [18], issues that faculty consider when examining new and innovative teaching materials for adoption include: preparation time, material contents, departmental oversight committee, and compatibility of programming languages. Adopting/adapting results from an extensive games development approach requires a significant investment of time which includes faculty understanding a game engine or significantly reworking existing curriculum. This work intensive adoption/adaptation is not suitable for limited scope investigation, especially for faculty members with no backgrounds in computer graphics and games. Projects and results from the per-assignment games development approach are typically from faculty members with expertise in graphics/games and are "*student-centric*" where the main goals of study are student engagement and various learning outcomes. Most instructors of CS1/2 courses do not have the time or expertise to adapt and/or implement these projects in their courses.

The GTA modules are limited curriculum scope, self-contained CS1/2 assignments, that require no existing knowledge of games or graphics from the faculty and require minimum changes to existing classes in order to be adopted. These modules are "*student-centric*" because they allow students to practice CS concepts in context. More importantly, these modules are "*faculty-centric*" because they are the stepping stones for faculty to begin experimenting with a promising new approach to teaching CS1/2 courses.

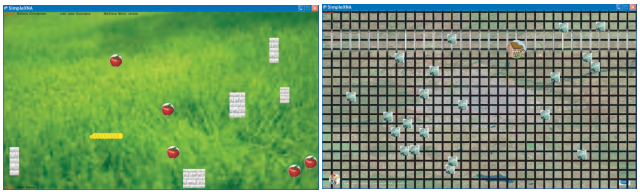

Figure 1: Games by Faculty with no prior backgrounds.

3. GTA WORKSHOPS FOR FACULTY

Outside of classrooms, GTA-related workshops had been offered for interested faculty members at regional and national (e.g., [21]) conferences, and at institutions internationally (e.g., [22]).[3] These workshops guided faculty to develop game-themed applications based on tutorials from the GTA modules. Images in Figure 1 are screen shots of a "*Worm-like*" and a "*Pizza Delivery*" games resulting from the second day of multi-day version of the workshops (e.g., [22]). Although relatively simple, these games were designed and developed in a matter of hours by CS1/2 faculty members with no prior background in graphics/games.

The workshops have received overwhelmingly positive feedback from the participants. For example, written feedback on "*the appropriateness of material difficulty*," included: "*I found the materials challenging but able to comprehend*," and "*good balance of complexity and new materials.*" While the written feedback on "*the presented materials will help me develop game-themed applications*," included: "*For sure!*" and "*In this environment [*GTA*], YES!.*" The results and feedback from these workshops showed that, though they found the GTA materials to be non-trivial, faculty participants with no prior expertise in graphics/games were able to comprehend and begin developing game-like applications within a matter of hours.

4. ASSESSMENT TOOLS AND PROCEDURES

Based on the positive feedback and promising outcomes from the GTA workshops, we began in-classroom adoption of GTA modules. Because the modules are designed as catalysts for faculty development, faculty members needed to have the reassurance of consistent student learning outcomes when experimenting this potentially powerful approach to teaching. This project utilized exam scores, qualitative evaluation of projects, student attitude survey, and success rates [7, 9] to assess student learning outcomes and perceptions when the GTA modules were adopted in existing classes.

For long-term effects, new teaching materials are tracked over multiple semesters of the same course via course enrollments (e.g., [19]) or continual student successes (e.g., [3]). In the case of GTA modules, one goal was to demonstrate the feasibility of simple replacement of selective assignments in existing classes. The same classes were followed over different semesters where different GTA modules were adopted. The assessment procedure was designed around existing *console-based*[4]CS1/2 classes.[5] These were well established courses producing many successful alumni in advanced CS courses

[3]Refer to *http://faculty.washington.edu/ksung* for workshop lecture notes.

[4]*Console-based*" or "*console assignments*" refer to programming assignments based on keyboard and character driven console monitors. A "*console course*" is a course based on console assignments.

[5]The involved instructor has no background in graphics/games.

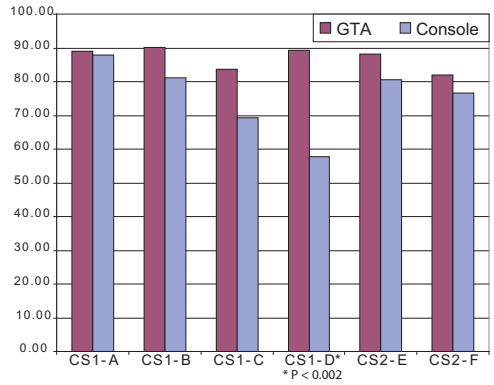

Figure 2: Average Assignment Results.

and in the industry [1].

Each GTA module included a non-game console assignment and a game-themed assignment. To facilitate the assessment procedure, in phase one of the GTA project the console versions of the assignment was designed to be technically equivalent to the ones in existing CS1/2 courses[6]. In this way, the GTA replaced corresponding assignments without incurring any changes to the courses.

Of the six GTA modules developed thus far, four were targeted for CS1, and the other two were designed for CS2. The game-themed versions of these assignments were integrated into existing CS1/2 courses over three academic quarters. Two existing CS1/2 courses were offered without modification and served as control groups. In the experimental classes, two of the four existing console assignments were replaced with GTAs. Each GTA course consisted of a mixture of two existing console and two GTA modules in combinations which varied from class to class. This verified that a faculty member could select and replace some or all existing assignments with GTAs. To minimize variation between students in the classes, the modifications to the courses were not advertised, thus students did not know they were registering for a class using the GTAs.

5. RESULTS AND ANALYSIS

Throughout the experiment the instructor *avoided* any extra effort when adopting the GTA modules. For example, no lecture time was spent covering graphics or games aspects of the GTAs. In all cases, the *exact* same assessment instruments were administered under similar conditions for both GTA and console courses.

Success Rates

The success rates of these CS1/2 classes have fluctuated between 65% to 85% historically. In addition, because we do not offer CS2 in the Winter quarter, the Spring-CS2 class always has a higher enrollment than the Fall offering. With this historic perspective in mind, we began our analysis by examining the overall success rates of all the classes:

	Pass	Fail	Drop
CS1-F(G)	13 (72%)	5 (28%)	3
CS1-S(G)	13 (76%)	4 (24%)	2
CS1-W(C)	13 (65%)	7 (35%)	4
CS2-S(G)	18 (86%)	3 (14%)	2
CS2-F(C)	11 (79%)	3 (21%)	2

In the above table, the left column indicated the class (*CS1* or *2*), academic quarter (e.g., F for Fall) and types of class ("*G*" and "*C*" for GTA and console). The next columns show number and percentage that passed, failed, or dropped. The first three rows were results from the CS1 classes and the bottom two were from the CS2 classes. If examined closely, the *Pass* averages of GTA classes (72% and 76% for CS1 and 86% for CS2) were higher than the console classes (65% for CS1 and 79% for CS2). Caution should be used when examining these figures since they are well within the historic ranges. Because of the small number of female students, to ensure strictest anonymity, we did not analyze the results from the these students separately. The above analysis includes all of the students in our classes.

Assignment Scores

Figure 2 plots the average scores on assignments for all of the six assignments.[7] The left bar above each assignment displays the results from GTA while the right is from the console assignment. The scores from GTA were consistently better than the corresponding console version. From the written feedback (to be detailed later), it was clear that students spent more time *playing* with the GTA assignments and thus resulted in smaller number of errors in their final submissions. In addition, we observed an interesting *trend* in the score differences. At the very beginning of CS1 when the assignment was trivial (CS1-A, simple arithmetic), the difference between GTA and console were small. By the end of CS1, the assignments became more challenging (CS1-D, 2D array), and the difference in scores also became more prominent (89% vs. 58%). As widely recognized (e.g., [6]), interactive graphical feedback encourages experimentation and this trend in assignment score differences reflected students' further engagement with the assignment. However, this increase in score differences became less prominent by CS2 assignments (88% vs. 81% for CS2E, and 82% vs. 77% for CS2F). As observed by Guzdial [10], CS2 students are more experienced and often prefer assignments without elaborate setups. Our results verified this observation that as students became more proficient, the advantage of interactive graphical feedback diminished.

Self-Reported Time Spent on Assignments

Figure 3 shows students self-reported time spent on the assignments. When we looked at the time spent on the four CS1 assignments, it was interesting to note that students reported more than twice the amount of time spent on CS1-B, C and D console assignments. From the average scores and success rates, we know students were learning comparable materials. These large discrepancies seemed to suggest *game-themed* assignments were much more efficient learning tools. While it may be true that visual feedback is an important learning tool, we do not believe that alone it could

[6]The *technical equivalency* of the game-themed and console versions of the assignments has been verified by an independent faculty evaluator [24]. In the rest of this paper, *GTA* refers to the actual game-themed assignment, while *GTA module* refers to the entire collection of materials. Each GTA module contains a game-themed assignment (GTA) and a technically equivalent console version of the assignment.

[7]Note that each assignment has a GTA and console versions. For example, the GTA version of *CS1-A* was offered during Spring while the corresponding console version was offered in the control course, Winter.

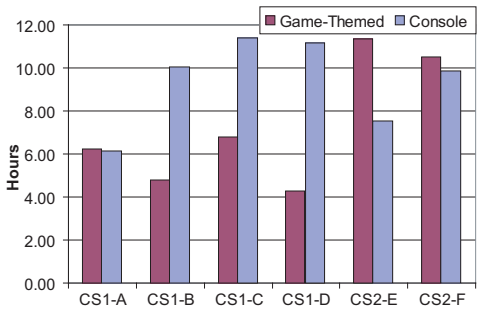

Figure 3: Self Reported Time Spent.

accomplish such impressive results. From the written feedback (e.g., "*counting only the time I actually "worked" on and not "played" with it*"), it appeared that some students discounted the time they spent *playing* with the sample solution and starter project. We believe the correct numbers for GTA should be closer to those from console assignments. In the case of CS2, the large time difference between the first GTA (CS2-E) reflected the fact that the assignment was more complicated and demanded time for familiarization. The similar amount of time spent for the second CS2 assignment (CS2-F) showed that students were able to take advantage of their initial time investment.

Per-Assignment Survey

After students handed in their assignments and before they received their grades, the exact same questions were asked to all students: *clarity* and *difficulty* of the assignment, amount they have *learned* from the assignment, if completing the assignment made them feel *satisfied*, and if the assignment was *interesting*.

Figures 4 and 5 represent the average of all results from GTA and console courses for all the CS1 and CS2 courses, respectively. Results in Figure 5 showed CS2 students found the console assignments easier to understand, slightly more challenging, and contrary to intuition, they found working on console assignments more satisfying and that the console assignments were more interesting.

Fortunately, written feedback from students helped explain the above observations. In both CS1 and CS2 GTA courses, students complained that they had to spend extra time understanding the given GTA starter project. However, in all cases, once they understood the given system, students reflected that completing the assignment was not "*as complicated as it first appears.*" In CS2 assignments, many students expressed frustration at not able to *improve* on the assignments (e.g.,"*it was interesting to have an end result you can play with, I wish I was able to improve the boring game*," or "*I did learn a lot about BST [Binary Search Tree] doing this assignment. However the 'game' parts were useless as we don't actually get to do gaming.*").

The GTA modules were designed to be *expertise neutral* to prevent superfluous graphics user interaction programming and inappropriate gaming contents (e.g., violence) [25, 12]. We have made a conscious decision to restrict students' access to such functionality [24]. This feedback showed that we were successful, however, the feedbacks also told us that we had literally taken "the fun" out of games programming. We are currently investigating the delicate balance between allowing creativity and discouraging excessive graphics programming.

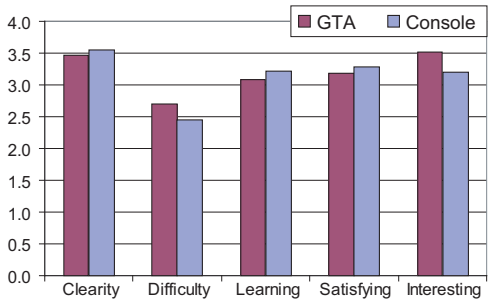

Figure 4: CS1 Post Assignment Survey.

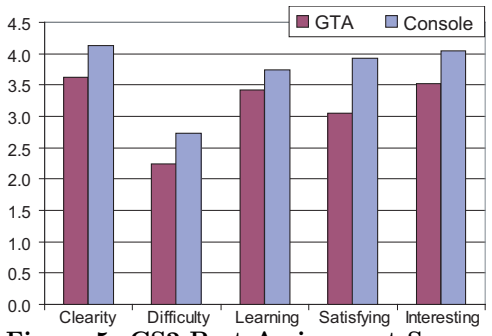

Figure 5: CS2 Post Assignment Survey.

Pre- and Post-Course Survey

We designed course survey forms to understand students' background, self-perception, interests, and general attitudes towards the CS discipline. Students completed these forms during first day of class (*pre*-course) and before the final exams (*post*-course). Tables 1 and 2 show the averages of all students from the *GTA* and *Console* classes separately.[8]

It is interesting that replacement of *some* assignments in our CS1/2 classes was almost *transparent*. The only exception was the "*Well prepared*" question for CS1 students. In this case, after the class when asked about "*in hindsight, how well do you think you are prepared for the class*," there was an obvious spike in self-confidence for GTA students. These data, in combination with the better performance results from GTA (e.g., CS1-C and D in Figure 2) and the indifference attitude from Figure 4, implied an interesting and potentially important observation. CS1 students performed better because the interactive graphical application supported experimentation and visualization. Since the applications were not really fun or flashy they did not find the assignments especially interesting. However, after the class, they did become more confident about their abilities. Together these data suggested that targeted "*uninteresting*" interactive graphical assignments can be a good tool for teaching CS1 students.

Feedback From Faculty

After grading each assignment the instructor filled out a feedback form detailing his efforts specific to the assignments (e.g., lecture time, answer questions, grading time), and impressions on student learning. As mentioned, the instructor did not have prior background in computer graphics/games and purposely avoided specific lecture time and extra help for GTAs. As a direct result, his feedback showed no significant difference in efforts between the GTA and console assignments. However, he reported students' verbal com-

[8]Students had the option to not participate in these surveys.

		Like CS	Career	Coding	Prepared	Difficulty	Scope
GTA (N=22)	Pre	4.50	4.64	3.91	2.90	2.61	3.41
	Post	4.50	4.68	3.68	3.86	3.41	3.77
	Change	**0.00**	**0.05**	**-0.23**	**0.96***	**0.80**	**0.36**
Console (N=10)	Pre	4.17	4.73	3.65	3.04	3.09	3.41
	Post	4.40	4.70	3.20	3.10	3.80	3.50
	Change	**0.23**	**-0.03**	**-0.45**	**0.06**	**0.71**	**0.09**

*P < 0.02

Table 1: CS1 Pre- Post- Course Survey.

		Like CS	Career	Coding	Prepared	Difficulty	Scope
GTA (N=18)	Pre	4.64	4.82	3.86	4.27	2.64	3.59
	Post	4.61	4.94	3.89	4.56	3.06	4.11
	Change	**-0.03**	**0.13**	**0.03**	**0.28**	**0.42**	**0.52**
Console (N=9)	Pre	4.54	4.38	3.46	3.54	3.08	3.69
	Post	4.44	4.78	3.67	4.11	3.44	3.78
	Change	**-0.09**	**0.39**	**0.21**	**0.57**	**0.37**	**0.09**

Table 2: CS2 Pre- Post- Course Survey.

ments on GTAs to be more work and more difficult[9]. In the end, because the GTAs were "*dropped*" into the classes without any dedicated lecture time, the assignments had minimal effect on the class as a whole. It is encouraging that as the instructor became more comfortable with GTAs, he did begin experimenting with graphics/game programming and developed a simple card matching game based on the provided tutorials. Currently, the instructor is experimenting with incorporating game-themed instructional modules in his CS1 classes [2].

6. CONCLUSION

The resulting games and survey feedback from GTA workshop participants indicated that it is straightforward for faculty without graphics/games background to understand and begin working with GTA modules. Our classroom case study demonstrated that it is possible for a CS faculty member with no background in graphics/games to integrate GTA modules in an existing course without adverse effects on student learning. These results are exciting because interested faculty can confidently begin limited curriculum scope experimentation with selected GTA modules in their own courses. To further support these faculty members, we have developed limited curriculum scope, self-contained "Game-Themed Instructional" (GTI) modules for teaching individual programming concepts (e.g., linked-lists, or arrays) [2]. We envision interested faculty members experimenting with selected GTA and GTI modules in their existing classes, becoming comfortable with game-themed teaching approach, consulting the tutorials provided with GTA and GTI modules, and beginning to develop their own game-themed materials. We believe the true potential of engaging and exciting students can only be realized when the instructor becomes proficient in and feels ownership of the instructional materials [4].

Currently, we are designing a multilingual and API-independent platform to support GTA/GTI modules in multiple programming languages and APIs. In addition, we are working on sharing our results with colleagues from community colleges and high schools and continuing to offer workshops for interested faculty members.

7. REFERENCES

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

[2] R. Angotti, C. Hillyard, K. Marino, M. Panitz, K. Sung, and J. Nordlinger. Game-themed instructional modules: A video case study. November 2009. Work in progress.

[3] J. D. Bayliss. The effects of games in cs1-3. *Journal of Game Development*, 2(2), 2007.

[4] J. D. Bayliss. Using games in introductory courses: tips from the trenches. In *SIGCSE '09*, PP. 337–341, 2009.

[5] 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*, PP. 354–358, 2006.

[6] S. Cooper, W. Dann, and R. Pausch. Teaching objects-first in introductory computer science. In *SIGCSE '03*, 2003.

[7] J. Cromack and W. Savenye. Learning about learning in STEM education, 2007. *http://research.microsoft.com/ur/us/AssessmentToolkit/*.

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

[9] M. Eagle and T. Barnes. Experimental evaluation of an educational game for improved learning in introductory computing. In *SIGCSE '09*, PP. 321–325, 2009.

[10] M. Guzdial. Contextulized computing education. In *Invited Presentation, Microsoft Research Faculty Summit, http://home.cc.gatech.edu/guzdial/169*, July 2008.

[11] P. Haden. The incredible rainbow spitting chicken: teaching traditional programming skills through games programming. In *ACE '06*, PP. 81–89, 2006.

[12] S. Haller, B. Ladd, S. Leutenegger, J. Nordlinger, J. Paul, H. Walker, and C. Zander. Games: good/evil. In *SIGCSE '08*, PP. 219–220, 2008.

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

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

[15] R. B.-B. Levy and M. Ben-Ari. We work so hard and they don't use it: acceptance of software tools by teachers. *SIGCSE Bull.*, 39(3):246–250, 2007.

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

[17] A. Luxton-Reilly and P. Denny. A simple framework for interactive games in cs1. In *SIGCSE '09*, PP. 216–220, 2009.

[18] L. Ni. What makes cs teachers change?: factors influencing cs teachers' adoption of curriculum innovations. In *SIGCSE '09*, PP. 544–548, 2009.

[19] I. Parberry, T. Roden, and M. B. Kazemzadeh. Experience with an industry-driven capstone course on game programming. In *SIGCSE '05*, PP. 91–95, 2005.

[20] K. Sung. Computer games and traditional computer science courses. *Communications of the ACM*, 52(12):74–78, December 2009.

[21] K. Sung. Xna game-themed applications for teaching introductory programming courses. *Invited Pre-Conference Workshop, FDG*, April 2009.

[22] K. Sung. Xna game-themed applications for teaching introductory programming courses. *Invited 3-Day Workshop, Digital Arts University, Mexico*, February 2009.

[23] K. Sung, M. Panitz, R. Reed-Rosenberg, and R. Anderson. Cs1/2 game-themed programming assignments for faculty. *Journal of Game Development*, 3:27–47, March 2008.

[24] K. Sung, M. Panitz, S. Wallace, R. Anderson, and J. Nordlinger. Game-themed programming assignments: the faculty perspective. In *SIGCSE '08*, PP. 300–304, 2008.

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

[9]The instructor did not have access to any survey information during the classes.