Teaching teamwork and communication skills by using a studio-based learning model in a multidisciplinary course on game design

by

Anthony Estey
Bachelor of Science, University of Victoria, 2008

A Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

in the Department of Computer Science.

Teaching teamwork and communication skills by using a studio-based learning model in a multidisciplinary course on game design

by

Anthony Estey
Bachelor of Science, University of Victoria, 2008

ABSTRACT

Jobs in the computing field demand communication and teamwork skills in addition to programming skills. Focus needs to be shifted at the undergraduate level towards developing collaborative skills to enable a smooth transition into employment in academia or industry. With computer science bachelors degree production at a record low, games courses have been gaining in popularity, as there has been growing evidence showing positive enrollment and student engagement results. Building upon other game programs that had successful results, I present a game design course developed to attract students of all disciplines. Our course is different because we focus on three main issues identified by recent industry studies: cooperative learning, peer review, and team orientation. The course was successful in attracting students across multiple disciplines, and an analysis indicated increased student interest in pursuing a computer science degree. Unfortunately, the same pre- and post-surveys suggested that our collaborative activities may have resulted in a decrease in student interest regarding course work and in pursuing studies in game design. Student feedback also informed us that students felt uncomfortable participating in some of our peer review activities. Because of these results, I used a studio-based pedagogical approach to restructure the peer review activities in our course. In our previous offering, students received peer feedback only on their final game presentation. In our latest offering, we integrated peer review activities into every step of the game development process, allowing students to use the feedback received in the peer review activities to refine their work while progressing through a game project. A quantitative analysis informs us that our refined peer review activities were successful in increasing student presentation confidence, sense of competition and community, and excitement towards their course projects. A qualitative analysis suggests that studio-based learning can provide a better learning environment for our students. Most importantly, students reported that they found this type of peer review to be useful in aiding them towards achieving their learning goals. The course at the University of Victoria is a course on game design, but I suggest that the studio-based learning model can be very effective in any course where students are instructed to submit unique projects, or unique solutions to a given problem. I recommend studio-based learning to any educators interested in cooperative learning, or considering integrating peer review activities into their course work.

Contents

Supervisory Committee ii

Abstract iii

Table of Contents v

List of Tables vii

List of Figures viii

Acknowledgements ix

Dedication x

1 Introduction 1

1.1 Overview 5

2 Background and Related Work 7

2.1 Games 7

2.1.1 Multidisciplinary games course 9

2.2 Cooperative learning 10

2.2.1 Presentations and Peer Review 12

2.2.2 Studio-Based Learning 15

2.2.3 Studio-Based Learning and Game Design 16

3 Course Design 17

3.1 Fundamental Design Decisions 17

3.1.1 Games course 17

3.1.2 Cooperative learning 19

3.1.3 Peer Review 21

3.2 Course Outline 25
3.2.1 Learning Objectives and Assessment 25
3.2.2 Assignments 25
3.2.3 Quizzes, Labs, and Textbooks 26
3.2.4 Group Organization 27
3.3 Addressing Industry Issues 27
3.3.1 Cooperative learning 28
3.3.2 Peer Review 28
3.3.3 Orientation with a Pre-Existing Code Base 29
3.4 Changes made for the Fall 2009 offering 30
3.4.1 Peer Review Redone 32

4 Analysis: Fall 2008 Offering 35
4.1 Quantitative Results 35
4.1.1 Survey Significance 38
4.1.2 Comparison: Games vs. Traditional 39
4.2 Qualitative Results 42
4.2.1 Cooperative Learning 42
4.2.2 Peer Review 43
4.2.3 Orientation with Pre-Existing Code Base 43

5 Analysis: Fall 2009 Offering 45
5.1 Quantitative Results 45
5.1.1 Survey significance 58
5.2 Qualitative Results 59

6 Observations and Discussion 61
6.1 Fall 2008 offering 61
6.2 Fall 2009 offering 62

7 Conclusions 66

Bibliography 68

List of Tables

Table 3.1 Project milestones involving peer review 33

Table 4.1 Interest in pursuing a degree in CS 36
Table 4.2 Likelihood to enroll in another CS course 36
Table 4.3 Interest in further developing programming skills 37
Table 4.4 Interest in learning about game design 37
Table 4.5 Interest in a more advanced course on game design 38

List of Figures

Figure 4.1 Change of interest for CS majors in CS-games 39
Figure 4.2 Change of interest for CS majors in CS-traditional 40
Figure 4.3 Change of interest for non-CS majors in CS-games (26 students) 41

Figure 5.1 Student desire for more peer review activities 46
Figure 5.2 The effects of peer review on sense of competition 47
Figure 5.3 Peer review effect on motivation to participate in class discussion 48
Figure 5.4 Effects of peer review on excitement towards coursework 49
Figure 5.5 Degree in which students felt peer review could have improved earlier assignments 50
Figure 5.6 Effects of peer review on student sense of community 51
Figure 5.7 Degree to which students felt peer review helped them learn new ideas about coursework 52
Figure 5.8 Effect of training excercises on student presentation confidence 53
Figure 5.9 Effects that demo peer review sessions helped students 54
Figure 5.10Student confidence towards giving peer review 55
Figure 5.11Effects of peer review on motivation to succeed 56
Figure 5.12Effects of peer review on self-assessment 57

ACKNOWLEDGEMENTS

I would like to thank:

Bruce & Amy Gooch for believing, and teaching me how to begin writing a paper,

Yvonne Coady for leading me by the hand through the following steps.

Celina Gibbs for the unbelievable guidance, feedback, and support.

Jeremy Long for the kind and constant help and advice on a daily basis,

Sven Olsen for the long philosophical talks that re-sharpened my focus.

My wonderful friends for making every day of my life in Victoria so enjoyable,

And my loving family for the never-ending support in all of my endeavours.

DEDICATION

Brenna Innes
You are the inspiration behind my success

Chapter 1

Introduction

"To draw students to CS, we must first look to create a curriculum that reflects the exciting opportunities and challenges of IT today versus the 1970s. Future students and faculty would greatly benefit from a reinvigorated CS curriculum." David A. Patterson [49]

After the decline of enrollments in Computer Science (CS) programs over the last decade, many institutions have made significant changes to introductory courses or even full program curriculums. The percentage of incoming undergraduates among all degree-granting institutions who indicated they would major in CS declined by 70 percent between the Fall of 2000 and 2005, and there was further decline in Fall 2006 [60]. The efforts of many of these institutions may be paying off, as in a study given out in May 2009, it was reported that in the previous three years there was a 9.4 percent cumulative increase in the number of students per CS department [65]. Overall, student enrollments are still down since the turn of the century, so it is important that continued effort is focused on stimulating interest in computer science programs.

Student retention is of equal importance, as reports show that CS Bachelor's degree production has been on a steady decrease between 2003 and 2008 [64]. Bachelor's degree production in 2008 was down nearly twenty percent, and there was also a decline of ten percent in 2009, resulting in the smallest graduating computer science class in ten years [65].

The decline in bachelor's degree production is a major problem when we consider that careers in computer science are increasingly in high demand and continue to appear in top ten job opportunity lists. In a study done in 2008 on careers with respect to the highest combined scores for pay, projected workforce growth, and number of openings, computer science related careers fell into the top three overall positions [56]. Studies done in 2009 found similar results for computer science related careers, as reported by Wellman et. al [63]:

> “Not only are these jobs identified as currently in demand and recession proof, but the demand for these positions is predicted to continue to increase in the future.”

Computer science careers continue to top lists right into 2010, as there were three careers listed in Career Explorer’s Ten Hottest Careers: 2010 [28].

Some institutions have reported promising enrollment and retention results when incorporating a game-themed course into their CS program at a first or second year level [42, 17, 7, 48, 47]. There has also been promising work done by institutions that have studied the approach where students play games to teach computer science and programming concepts [5, 30, 23, 24, 59]. When given a choice, students preferred learning programming concepts through games [24], and liked game-based assignments over more traditional programming assignments [19, 18].

Previous work suggests that not only do students like learning about introductory programming concepts within the context of games, but also that games do not negatively affect student performance [7, 19]. Survey results at the University of Denver regarding the effectiveness of games in teaching computer science concepts reported that students did learn effectively [42], and that there were improvements in programming knowledge, as well as student confidence [2]. They recommend that the game approach should be seriously considered, as it is both compelling and fun for the students. These results provide some solid evidence that game-themed courses and assignments are very easy to substitute into a computer science curriculum.

Games are most commonly created in teams, which is an important aspect educators should consider when integrating game-focused courses into their curriculum. Currently, university students are not acquiring sufficient communication and teamwork skills to be comfortable or effective when they enter industry [1, 10, 11]. Industry studies have found that although university computer science curricula provide new software developers with adequate design and development skills, it is their communication, collaboration, and orientation skills that are not well addressed [10, 44, 11].

Pulimood and Wolz, at the College of New Jersey, claim that an authentic learning environment in which students solve real problems as a collaborative community with their peers from other disciplines is needed, and report on some positive results [51]. Ramsden asserts that good teaching should actively involve students in the learning process, and should promote independence and a sense of student control over their own learning [52]. Dochy et al. state that peer assessment improves different aspects of the quality of learning of students, and that above all, the main reason peer assessment needs to be integrated in curricula in higher education is its impact on the learning process [29]. Blume et al., at the University of Toronto, have introduced a communication skills elective course into their computer science program, which integrates instruction in written, oral, and interpersonal communications [13]. Blume states that computer science (CS) departments must first acknowledge the importance of these skills and realize that great improvements can be made to students communication abilities. English professors rarely teach technical writing, and public speaking courses are often relegated to continuing studies departments or community colleges. Thus, it is recommended that computer science departments should provide activities to address these types of skills and student experiences internally.

Courses designed to help develop the communication and teamworking skills of students yields further benefits to students. A study done at the University of British Columbia reported that team-based learning improved interest in course material as well as student confidence [40]. A common concern is that it is difficult to seamlessly integrate these activities into course work, but results from a study done at San Diego State University show that the benefits of cooperative learning clearly outweighed any possible losses due to reduced instructor lecture time [8]. Studies on cooperative learning in CS1 courses have also provided positive evidence that these types of activities continue to provide benefits to students as they go on to CS2 courses [8, 31]

Here at the University of Victoria, we created a multidisciplinary computer science game course, introducing students to many core Computer Science concepts in an engaging manner, which in our case was through the context of gaming [26]. One of our goals was to emulate experiences our students might find themselves facing if they were to pursue a career in industry. Thus, we introduced new instruction and assessment methods that align directly with concerns recently discussed in the aforementioned [11, 44] industry studies:

1. Cooperative learning. Cross-discipline group work.
2. Peer review. Give and receive constructive critiques.
3. Orientation with a pre-existing large code base.

This course was first piloted in the Fall 2008 semester. Although we were successful in providing students with some domain experience, student feedback revealed that students were uncomfortable performing our collaborative learning tasks, especially those revolving around peer review. Particularly concerning was the fact that survey results showed a drop in student interest in regards to course work and in further pursuing studies in game design. Other researchers addressing communication and teamwork skills have also found that students and instructors alike perceive that the students made significant improvements in communication, presentation, and teaming skills, but that students were extremely uncomfortable with the paradigm shift in their learning environment [38]. These results prompted us to investigate a way to refine our course's teamworking activities in a way to promote student enthusiasm towards coursework and confidence in a group setting.

Some institutions have explored using a pedagogical approach called studio-based learning [36], in which students present solutions to both their instructors and peers for feedback and discussion during different benchmarks in every assignment. This allows active discussion of algorithms and implementation methods during assignments, allowing students to share and discuss multiple ways of solving a given problem. By providing students with feedback before the final submission of an assignment, students are provided with an opportunity to apply the suggestions received by their assessors, and explore multiple solutions to a given problem. When students provide feedback to their peers it also prompts self reflection, and gives them the ability to fill roles typically reserved for course instructors. These institutions have had success integrating peer review activities into computer science courses; they were able to increase student engagement with respect to coursework, while providing students with some communication and teamworking experience [35, 45]. Formative assessment, where students are provided with feedback as they progress through an assignment, as opposed to summative assessment, where they receive feedback only after being graded on a completed assignment, can potentially greatly increase learning outcomes [14, 46]. This instant feedback on assignment work is very valuable to students, as reported by Søndergaard, at the University of Melbourne [57]:

> "It seems that, if there is one thing educational experts can agree on, it is the crucial importance of timely, helpful feedback on student work. Yet in almost every class, in almost every tertiary education institution, feedback remains the weak point in surveys of student satisfaction."

We restructured the peer review activities for the Fall 2009 offering of our course, and investigated the effectiveness of using a studio-based learning approach in a course focused on game design [27]. A quantitative survey provided us with results on some of the positive influences the peer review activities had on our students:

- Sense of community and competition.
- Motivation and excitement towards class performance.
- Perceived value and applicability of peer feedback.

Based largely on the very positive anecdotal evidence we found with respect to our students in combination with the qualitative survey results, I feel our games course really benefited from the integration of the studio-based learning approach. The main contributions of this work should be particularly appealing to educators interested in either cooperative or game-focused learning. In the following chapters I suggest the important and fundamental factors an educator must consider in the design of either a course on game design, or a course focused on cooperative learning. I report on the two offerings piloted at the University of Victoria, and the important changes I made to certain activities between the offerings in order to address student and educational concerns. I provide an analysis on survey results, student feedback, and observations make throughout both offerings, and conclude with a discussion on why I feel a games course is the ideal place to maximize the benefits a studio-based learning approach provides.

1.1 Overview

Chapter two of this thesis begins by discussing related and previous work in the relevant areas of game-focused computer science courses, cooperative learning, and more specifically peer review and student presentations. In chapter two the studio-based learning approach is also further explained. I describe how a collection of related works provided a solid framework from which we were able to design the Fall 2008 offering of the course, and how recent studies done at other institutions were influential to us when we refined our course for the Fall 2009 offering. The main contributions of this work stem, in part from the thorough case studies and reports published by other institutions, as well as the feedback and results returned by the students throughout the two offerings of our games course here at the University of Victoria, discussed in later chapters.

Chapter three focuses on the design of our course, and I highlight how our course was designed simultaneously with a more traditional programming course, where both courses were mapped to cover the same learning outcomes. I overview the fundamental design decisions an educator must consider when interested in the creation of a games course. I also look at the important factors to consider when trying to create a cooperative learning environment, and then overview key principles to consider in order to maximize the effectiveness of peer review. Chapter 3 also outlines our grading rubrics, number and types of assignments and quizzes, and explains how we integrated the activities focusing on teamwork and communication into our course. It also explains the main changes between the Fall 2008 and Fall 2009 courses.

Chapter four is an overview of the analysis we performed on the Fall 2008 offering of the course, and specifically compares our results with the traditional computer science course also offered in the Fall 2008 semester. I share the results of the pre- and post-surveys regarding the course's impact on interest level, and also provide some qualitative analyses.

Chapter five provides a similar overview on the Fall 2009 offering. The results of a quantitative analysis of student responses submitted during an exit survey are presented. Qualitatively, I discuss some of the feedback we received from the students during this second offering of the games course.

In chapter six we overview the observations collected during both offerings of the course, and discuss some of these observations with respect to the survey questions and desired educational and learning outcomes.

In the final chapter of this thesis, I formulate conclusions and review the main contributions of this work. I also discuss some ideas and suggestions for future work with respect to games-based learning, cooperative learning, and higher level offerings of a multi-disciplinary game design courses.

Chapter 2

Background and Related Work

2.1 Games

Before the first offering of the games course at the University of Victoria, I undertook some rigorous review in an attempt to provide the best overall experience for our students, both in terms of student satisfaction and effectively providing them with some beneficial educational experiences. Games courses are certainly not a new idea, as institutions like the University of North Texas (UNT) have been offering games programming classes since 1993 [48]. To get some ideas about how to structure our course, we looked at games courses with a focus on collaborative learning, or multidisciplinary work. Courses that focused on aligning the course material with game industry standards were also of interest to us in our course design.

Ian Parberry, at UNT, has had success in both technical game programming courses for Computer Science majors, and game design and art courses for Visual Arts majors [47]. The programming class programmed the game engine, and the design and art class provided the art work for the game. In UNT's game programming courses, students were usually undergraduate seniors and experienced programmers, enabling them to commence with more advanced DirectX programming. The game design and art course focused on prototyping a system for action, and creating the environments and characters within the game.

I leveraged these successful results as my key inspiration in doing collaborative projects involving students of different technical backgrounds. Building from UNT's successes with collaborative game project work, the key significant difference in our course is that students of all majors are grouped into a single course. Additionally, our course does not require programming experience, so I did not consider it appropriate to start by programming in DirectX. We did, however, settle on a design where every facet of our course stresses collaborative learning, from labs to quizzes to major projects. Another difference is that we also heavily incorporated peer review into our labs, quizzes, and projects.

I also explored several of the courses in the University of Southern California (USC), undergraduate BS in Computer Science (Games) program [66]. The Game Design courses and the Game Cross-Disciplinary Courses were particularly interesting, as our course was also available to students of all disciplines. Their video game production class introduces students to all of the aspects of game development. The course brings in speakers from industry which they found helped retention, and built games individually using Game Maker[1]. The outline of this course was very similar to how we hoped to deliver our course. We also brought in speaker's from industry [44], and used Game Maker as our tool to build games. Our course is different because we stressed cooperative learning, instead of doing individual projects. All of our course activities were collaborative, multi-disciplinary efforts, and involved peer review.

Some universities, such as the University of Denver (DU), have focused on a Game Development undergraduate degree [42]. DU start with Flash to teach basic CS concepts, and move to C++ to strengthen the concepts. They base their language choices on their department standard and the demand of the game industry itself. Their CS1 courses are heavily programming oriented, and their results showed that students had fun while also learning effectively. Although our course was more of a general game design course, as opposed to our mainstream CS1 programming course, I was mindful of how they were successful in engaging students through interesting assignments.

Christopher Egert, at the Rochester Institute of Technology (RIT), developed courses to balance the technical and creative aspects of the curriculum [25]. They created three courses offered at the graduate level to expose students to game literacy, game design and creation, and the business aspects of game creation. After the course offerings, the author's found that students were more in tune with the interconnections between various media influences, and could quickly identify the need for graphics, audio, interactions, and story in a game. However, they found that students sometimes fail to realize that artists, modelers, audio engineers, musicians, Foley artists, user interface designers, writers, and programmers must all interact

[1]http://www.yoyogames.com/gamemaker

to create the experience. We are offering our course at the first-year level to all disciplines. With our collaborative team projects, I hoped students would realize how important, and sometimes challenging, communication is between the different production teams involved in the development of a game.

The positive results apparent in all of the aforementioned work gave me a solid framework to work with. I worked to build upon these ideas to design a course different from these offerings in two key ways. First, this was not a core CS1 course, so I could creatively alter the programming portion of the material. Second, my main focus was a game design course modeled around the teamwork skills industry has noted are lacking in new graduates: cooperative learning, peer review, and orientation with a pre-existing large code base.

2.1.1 Multidisciplinary games course

The course at UVic is different from other game courses as it is a multidisciplinary course heavily stressing collaborative work, group orientation, and peer review activities [26]. Art and writing students are not expected to be core team programmers, but must be able to effectively communicate with the programmers on their respective teams to succeed. For the second offering of the course, I restructured our peer review activities in an effort to stimulate further student engagement in course work.

In between the Fall 2008 and the Fall 2009 offering of our course, a piece of work was also published about a course at the University of Alberta (UA), which was also a multidisciplinary games course, heavily influenced by game industry partners [58]. Although the course at UA was offered at the second year level, their course was very similar to ours, and so provided some valuable information when restructuring some of our course activities. UA also provides their students with a collaborative environment that allows students to work on a project with peers of different disciplines. UA's approach is to impart these experiences through a game development cycle to provide students with a real-world learning experience. The students are provided with a set of tools and create a "module" for BioWare's game Neverwinter Nights[2]. UA researchers received positive feedback from most students regarding the course, but believe it was at least partially due to the students' passion for games in general. UA researchers did not evaluate the collaborative aspects of the course beyond the standard UA course evaluation.

[2]http://nwn.bioware.com

The game design course offered at UVic is distinguished from UA's course in several significant ways. First, the course we offer at UVic is at the *first year* level. Second, students are not provided with any tools to aid programming, or any multimedia assets; the programmers, artists, musicians, and writers on each team are responsible for creating all of their own work. UA provided their students with the Neverwinter Nights game engine, which includes a comprehensive multimedia collection. The activities in the Fall 2009 offering at UVic focused heavily on peer review, and although we do teach software engineering principles such as the game development cycle, instead of working on an engine of a pre-existing game, innovative game concept design is more heavily stressed. Furthermore, I specifically evaluated some of the effects the peer review activities present in our course have on our students in our exit survey.

2.2 Cooperative learning

Industry demand is not the only reason I chose to integrate cooperative learning activities into our course. Cooperative learning can influence students attitudes toward the course material in a way that advances them in Perry's scheme of intellectual development [50]. For example, by working on problems with peers, students may recognize a multitude of correct solutions to a given problem and the degrees of quality of each solution, rather than seeing every problem as having a right or wrong answer, with the right answer provided by the instructor. Cooperative learning activities can also be set up to help students reach all levels in Blooms Taxonomy [12].

Although many educators agree on the merit of integrating cooperative learning activities into computer science, team-based activities are commonly only used in senior level courses. Students should be taught how to work in a team, and how to effectively communicate with their peers early on in a degree, to make it a gradual process, not a sink or swim experience in the final year [55]. Hogan and Thomas warn that often we shortchange our students by giving them limited opportunities to learn how to work effectively in teams [34]. They report that students often complain that although they are expected to work in teams to complete projects, they are never given any advice or guidance on how to properly work in a team, and so in their first year courses they immediately focus on emphasizing a core set of skills related directly to working on teams. Hogan and Tomas' report was very helpful to me, as when designing the cooperative learning elements of our course, I was specifically interested in initiatives taken by other institutions at a first year level. I was also interested in recommended group sizes, types of activities, and most importantly, now to train students to effectively and comfortably work and learn together. Beck et al., at San Diego State University (SDSU), have done multiple studies on the effectiveness of cooperative learning in CS1 [8, 9]. Their work, along with a few other studies, was very helpful and influential when designing how to integrate the cooperative activities into our course.

SDSU developed a set of cooperative learning materials for a course at the CS1 level. Through a controlled experiment, they found that the cooperative learning experience had a significant positive effect on student performance [9]. Although the activities at SDSU were designed for a programming course, many of their fundamental ideas were applicable in our games course. Similar to SDSU, we elected to place our students into groups of between 4 and 5 students, allowed them to select different roles for different activities, and encouraged them to learn and progress through problems as a group. Our multidisciplinary course is offered at a first year level, and Beck et al. found that the benefits of cooperative learning were found by students from a variety of majors enrolled in their class [8].

Henry Walker, at Grinnel Colledge, states that changing to a collaborative learning approach requires a serious rethinking of the course, and can be a major adjustment to the instructor [61]. Walker reports that in a cooperative learning environment about 20% of the lectures present new material, while over 70% of the lectures are a mix of group exercises and discussion. Walker claims the change in teaching style encourages active learning, expands communication skills, covers more material better, increases retention rates, and supports alternative learning styles. Similar learning methods have been used at Sam Houston State University (SHSU), and the University of British Columbia (UBC). Both institutions adjusted their lecture format so new materials were presented by the instructor for only a small portion of total lecture time. At UBC, after an introductory lecture, students would then complete exercises and quizzes on the new material in teams [40]. At SHSU, students would work on exercises for the lecture in groups, and then be assigned work to complete for the next lecture, for which they present their experiments and results to the rest of the class [31]. Both institutions found very positive results regarding student performance, interest and confidence. Because of these positive results, I planned to incorporate group activities and quizzes into our course material to parallel weekly lecture material.

There were also some studies done that warn educators about some possible disadvantages to cooperative learning. Barker warned that simply requiring students to work in groups does not necessarily lead to improved learning outcomes [3]. Barker states that not only do instructors need to provide an environment where students are able to succeed with respect to Johnson and Johnson's requirements for successful student learning through collaboration (explained in section 3.1.2), but that instructors must also actively work with students in ways that will urge students to learn new skills rather than only improving their current ones. Barker et al., at the University of Colorado, also did a study on the reasons behind guarded behaviour and a defensive climate in the classroom, and the resulting negative effects on students [4]. The study reported that an impersonal environment, where it is easy to remain relatively anonymous and distant, is apparent in most computer science courses. The report stresses the importance of self-disclosure, even things as simple as name-swapping, to help cement interpersonal relationships and mutual trust. I believe cooperative learning to be an ideal way to better the classroom experience for our students, and took into consideration these issues when determining how to best deliver our course activities in our lectures and labs.

2.2.1 Presentations and Peer Review

When creating a class focused on cooperative learning, there are two very important choices an instructor has to make. First of all, the length and number of presentations each student will be responsible for throughout the semester should be decided. The second choice has to do with the types of feedback and evaluation the presenter will receive from the instructor(s) and the rest of the class during and/or after each presentation. Students need to be instructed on how to effectively give presentations, as well as how to effectively give *and* receive peer review.

There has been a lot of research in recent years on different types of peer review, and how it can be brought to bear in computer science education. Hamer et al., at the University of Auckland, reported on a study on over 1500 students using peer assessment in introductory programming courses [32]. They found that in engineering assignments, teaching assistants (TAs) out-perform peer reviewers with regards to Hamer et al.'s lexical sophistication metric, but in computer science assignments, this was not the case. In computer science assignments, peer reviewer marks were highly correlated with those of the TAs, and the need for TAs for a quality assurance measure was not strongly supported by their analysis. Reily et al. [53], also found that peer reviews were accurate when compared to an accepted evaluation standard, and that students actually preferred reviews from other students with less programming experience than themselves. Their results also suggest that students who participated in the peer review process achieved better learning outcomes.

Studies at the Harbin Institute of Technology found that peer code review is accepted as an ideal way to maximize the learning outcome of students in writing quality code, but there were some problems with the implementation of peer review, such as lack of qualification, and conspiracy activities [62], where students attempt to alter their peers' marks by providing dishonest feedback. Denning et al., at the University of California, San Diego, investigated tool support for "in-class lightweight preliminary peer review". Survey results showed that the peer review process caused students to reconsider their solutions, but that almost half of the students found it difficult to rate and classify their peers' solutions. Furthermore, almost half of the students felt some apprehension when participating in the peer review process [22]. Online peer review activities are an option, as Chinn had some success when he anonymously posted a select few solutions from each assignment to the web for critique [16]. Bauer et al., at the University of Vienna, studied the effectiveness of online peer reviews [6]. Students found that it was difficult to critique in a face-to-face setting, and that online reviews were much easier to do. The problem with online peer reviews is that there were a lot of misunderstandings that the students felt would not be apparent in a face-to-face setting. At the very least, student felt peer reviews would be much more effective if they were given the chance to communicate in person with their reviewers.

There has also been some recent research focusing particularly on student communication skills and presentations. Kaczmarczyk et al., report on a 10 week course offered to freshman, in which teams of students made between three and four 40 minute presentations to the rest of the class [38]. During the first of three weekly meetings, the instructor would introduce the weekly programming project, and in the following two meetings, students would present their work. During the first week, the instructor led a 30-minute discussion on how students could prepare and present their work. The instructor encouraged students to start with a short introductory lecture on important vocabulary and concepts, followed by a demonstration, and to conclude with a hands-on activity. Students and instructors alike perceive that the students made significant improvements in communication, presentation, and teaming skills and acquired deep content knowledge from their experience in the course. Pre- and post-surveys also show that students were uncomfortable with the shift in the learning environment, as it challenged their preconceived idea about how classroom instruction should operate. The University of Toronto (UT) offers a course on communication skills, in which students give six presentations, the first four ranging from 60-120 seconds, and the final two increasing to up to five minutes each [13]. During the first four presentations, the instructors would coach students on their presentations, and focus on points that would benefit the whole class. Students also provided feedback for their peers, based on pre-determined criteria for each different presentation. All presentations were also digitally recorded and then posted them online. Students rated the overall learning experience to be “high”, at approximately six on a seven-point scale. Unfortunately, the study at UT did not cover student satisfaction.

I took the aforementioned studies into consideration when rethinking the peer review and presentation activities for the second offering of our game design course. To counter the students’ perception of being unqualified, I looked at a study done at the University of Florida (UF) [20]. In an Artificial Intelligence in Computer Games course, peer evaluation was used along with teacher evaluations to grade student presentations. Lectures were given on how to give effective presentations, and how to provide constructive comments to each speaker so they could learn from their errors. Students were provided with an evaluation form to fill out, providing the rubrics for the peer evaluation. The peer evaluation activities at UF proved to be remarkably effective in identifying both faults and good traits within the student presentations. Unfortunately, there were no results on how the peer review activities actually affected student interest in the material. At Denison University, the Department of Mathematics and Computer Science have introduced significant new oral communication component early on in both majors [33]. They teach about qualities common to good talks during the first two weeks of the course. Students give three talks throughout the semester; the talks are five minutes, seven minutes, and ten minutes in length. Students are graded on preparation, delivery and visual media. Peers give critique, and are encouraged to give honest, but constructive criticism, and although peer reviews do not affect a speaker’s grade, the reviewers lose marks for every critique they do not submit. When restructuring the peer review activities in our course at the University of Victoria, I took into account that students must be provided with a set of rubrics on how to properly give peer review, and that it is also important that students are taught, and examples are given, of effective feedback. I also wanted to gradually introduce the peer review process, in a way as to promote student confidence and comfort in both giving and receiving peer reviews. I liked the idea that peer reviews do not affect a presenter's grade, but that reviewers are graded based on participation. I did not, however, want our presentation and peer review activities to happen only at the end of a project, but to integrate these activities into the course of each project, so they occured while students were progressing through each phase of their game's development. This pedagogical approach to cooperative learning is called studio-based learning.

2.2.2 Studio-Based Learning

Researchers at Washington State University, Auburn University, and the University of Hawaii, introduced an approach called studio-based learning [36]:

> "Adapted from architectural education, the studio-based instructional model emphasizes learning activities in which students (a) construct personalized solutions to assigned computing problems, and (b) present solutions to their instructors and peers for feedback and discussion within the context of "design crits" (design critiques). Thus, students participate in studio-based courses in ways typically reserved for instructors. Rather than emphasizing individual problem solving in isolation, studio-based courses regard group discussions mediated by student-constructed solutions as educationally valuable."

Hundhausen et al., at Washington State University, did an empirical study on peer review as a means of engaging students more actively in the learning process by using a studio-based approach they call pedagogical code reviews [35]. They found a positive trend with regard to the effects pedagogical code reviews have on the quality of students' code, and also on how effective the code reviews were in stimulating meaningful class discussion about helpful critiques of code. For future work, they plan to administer surveys to measure the students' sense of community that studio-based learning provides. Myneni et al., at Auburn University, used studio-based learning in a CS2 course [45]. Their preliminary findings hint at the potential of studio-based learning as an instructional approach that could potentially increase student motivation and interest in computer science. A studio-based approach was used in a course on 3D-modeling, design, and fabrication at University of North Carolina Asheville [54]. The material was a combination of computer science, engineering, and information technology, emphasizing creativity. Both instructors and students found the projects to be personally rewarding and pedagogically effective.

2.2.3 Studio-Based Learning and Game Design

In section 2.1 I overviewed how many institutions have deployed a games course into their curricula, and have benefited from positive results in terms of enrollment, retention, and student engagement. In section 2.2.2 I reviewed how other institutions have achieved positive results regarding student engagement when using a studio-based approach to give students communication and teamwork experience, although surveys regarding student satisfaction have yet to be administered. A studio-based pedagogical approach has been applied to programming and 3D-modeling design courses, but the effectiveness of applying the approach to a games course had yet to be studied and documented. I argue that a multi-disciplinary course on game design is a perfect fit for the team-based problem solving and peer reviewed activities apparent in a studio-based approach. I present a study on how effective the studio-based learning approach is in an introductory course on game design. I provide survey results on how comfortable students were in doing the peer review activities; whether it promoted self-reflection, as well as how it affected their sense of community, competition, motivation, class participation, level of excitement, and presentation confidence.

Chapter 3

Course Design

In the summer of 2008, Bruce Gooch, Amy Gooch, and I designed two first year introductory courses that complement the University of Victoria's CS1 programming course. For clarity, I will refer to these courses as *CS-games* and *CS-traditional*, respectively. During a workshop sponsored by the Learning and Teaching Center [41] at the University of Victoria, we determined learning outcomes, instructional strategies, and assessment techniques for the courses we were designing. The two courses were mapped directly to the same learning outcomes, with the CS-games content revolving around games, whereas the CS-traditional course had more conventional lectures and assignments. Both courses went through a preliminary review at the workshop, and both courses ran the same surveys throughout the semester. Between the Fall 2008 and Fall 2009 offerings of the course we made some significant changes to some of the activities in our course, specifically those involving peer review. I made changes to our course based on the results from the pre- and post-surveys and on student feedback. The changes I made, and the reasons behind our decisions, are explained in section 3.4. Chapter 4 compares and contrasts our survey results on student interest between CS-games and CS-traditional, respectively.

3.1 Fundamental Design Decisions

3.1.1 Games course

The University of North Texas (UNT) identified five key decision questions that affect the outcome of a games course in fundamental ways [48]. Here I present those questions and our corresponding responses.

Should the classes be theory based, or project based? Similar to UNT, we chose to design our class to be project-based, so the students would finish the course with playable products. Another factor in this decision was that course projects would play a very significant role in providing the experiences for our students with respect to developing their teamwork and communication skills.

What software tools should be used? We chose to use Moodle[1] as our course management system because it easily and effectively allowed us to place students into groups and create corresponding forums for the groups. Moodle also allowed us to create WIKI-based quizzes the whole class could easily partake in from anywhere a network connection was available. We chose to use Game Maker as the tool for the first game project, similar to USC's CSCI 280 Videogame Production course [66]. We felt its intuitive drag-and-drop nature would not overwhelm students without prior programming experience, and as the students become more comfortable programming, they could progress towards using the more advanced built-in programming language. As our course was not specifically a game programming course, but a course focusing on game design, we chose Inform[2], a design system based on natural language, for another assignment. We also chose Inform to accentuate the design process, story and rules, and to familiarize students with other factors important to the overall game development process.

Where should the students find art assets? As our class was not primarily a programming course, we encouraged students interested in the artistic elements of game design to choose to take the artist role for a game assignment, so for each game assignment one or more of the group members was responsible for providing the artwork for their group's game.

Should students be free to design any game in any genre, or should their choices be limited? We did not make any restrictions with regards to genre, so our students were able to make a game in any genre, as we wanted them to be able to work on games they were genuinely interested in. We did warn students that certain genres, such as first person shooters, could potentially require huge time investments for them to complete the development a working and playable product.

Should students write their own game engine, or work with a pre-existing engine? For the first game project, students worked in Game Maker. In the text-based adventure assignment, students worked in Inform. Neither required students to have any

[1]http://moodle.com
[2]http://www.inform-fiction.org

low-level programming knowledge. For the second game project, we gave students the option to create their own engine, or allowed them to work on an existing game engine we made available to the class.

3.1.2 Cooperative learning

In order to make the cooperative learning environment in our course as effective as possible, I looked at the *Characteristics of Effective Cooperative Learning*, which are the five characteristics Johnson and Johnson identify that make cooperative efforts more productive than individual efforts [37]:

Positive interdependence is the concept that members in a group perceive that they are linked with fellow group members in such a way that they can not succeed unless the rest of their group does (and vice versa), and/or that the whole group must coordinate their efforts in order to complete a task. Essentially, it is that students within a group feel that they all succeed or fail together, and that each member's efforts are both unique, required, and indespensable for group success. For our game projects, students chose different roles within their groups to fulfill throughout the course of the project. I believe this satisfies the requirements of positive interdependence, as no project could be complete without each group member completing his or her task. For example, if a game is programmed correctly but none of the art assets are completed, the game would not likely be considered a success, and would not receive a high grade.

Face-to-face Interactions may be defined as individuals encouraging and facilitating each other's efforts to achieve, complete tasks, and produce in order to reach the group's goals. Group members should provide each other with efficient and effective help and assistance, in regard to resources, information, insight, and motivation. Groups were formed by placing students in the same lab section together. Because of this, I was able to use lab sessions to facilitate many important group activities involving face to face interaction.

Individual Accountability and Responsibility. The purpose of cooperative learning is to make each member a stronger individual, and individual accountability is the key to ensuring that all group members are strengthened, and should afterwards be able to complete similar tasks by themselves. Individual accountability exists when the performance of individual students is assessed, the results are given back to the individual and the group, and the student is held responsible by group-mates for contributing his or her fair share to the group's success. It is important that the group knows who needs more assistance, support, and encouragement in completing the assignment. Although it is very hard to ensure students are being completely accountable for their work, we tried to introduce some activities to encourage student accountability. I informed students that after each group assignment there would be a peer evaluation activity where students evaluated other members of their group. During lab sessions, held once a week, groups were required to make progress reports for participation marks. As each student was responsible for a different part of their game, these progress reports were effective in motivating students to produce presentable work, and also revealed to groups which of their members may need assistance.

Social Skills. In order to coordinate efforts and achieve mutual goals, students must get to know each other, communicate accurately, accept and support each other, and resolve conflict constructively. Students must be taught the social skills required for high quality collaboration and be motivated to use them if cooperative groups are to be productive. We incorporated many activities to strengthen social skills into our lectures, labs, presentations, and even online quizzes.

Group Processing. The effectiveness of a group can be enhanced if students are given the time to review how effective their group was at achieving their goals. Students need to reflect on what group member actions were helpful and unhelpful, and make decisions on what actions should continue and which should change. This reflection enables groups to focus on maintaining good working relationships among members, facilitates the learning of cooperative skills, ensures members receive feedback on participation, ensures students think on both a cognitive and metacognitive level, and provides the means to celebrate the success of the group and reinforce the positive behaviours of group members. In a group reflection and evaluation activity distributed to the students after each game project there was a section where students were asked to fill out which actions performed by other members of their group they personally perceived to be the most helpful and which they felt were the most unhelpful. During the Fall 2009 offering, I created a class forum where students we required to post helpful feedback they received from their peers for participation marks. Students also presented game prototypes to the rest of the class, and during these game demonstrations it was immediately apparent which groups had succeeded in different aspects of their game's design. The prototype presentations gave group members time to reflect on what they were doing right, and what needed improvement, before having to submit the final versions of their games.

3.1.3 Peer Review

After the Fall 2008 offering of the games course, I wanted to restructure our peer review activities so that they were more effective and enjoyable for our students. In order to improve our peer review activities I had to determine what constitutes good assessment to begin with. In doing so, I would be able to maximize the benefits of our peer review activities. There have been many recent studies on peer review, but Nicol and Macfarlane-Dick's *Seven principles of good feedback practice* provide some key principles as well as specific strategies for instructors I felt were invaluable when considering how to restructure the peer review in our course [46]. I overview the seven principles of good feedback practice below:

Helps clarify what good performance is. Students can only achieve learning goals if they understand what the goals are, and are able to assess progress. If they are unable to to understand the assessment goals, the feedback they receive is unlikely to connect. Similarly, if students are unable to determine what good performance is, they simply will not be able to give effective feedback to their peers. Sometimes written assignment outlines are not enough, so discussions about desired learning goals are recommended. Demonstrating working examples to students is highly recommended, as it allows them to define a valid standard against which they can compare work. In our course, for the first assignment, students wrote a review about a game similar to one they would be making during the following two game projects. This provided them with a general idea of the quality and scope of the games they were expected to design themselves. In the Fall 2009 offering of the course, I was also able to demo a variety of games from the previous course offering, and explain the rationale behind how the games were evaluated.

Facilitates the development of self-assessment (reflection) in learning. When suitably organized, self-assessment can lead to significant enhancements in learning and achievement. Teachers need to create more structured opportunities for self-monitoring and judging of progression towards goals. Increasing the frequency of both subjective and objective review helps develop the skills which are transferred when students turn to producing and regulating their own work. During progress report activities I introduced into the lab sections, students were given feedback on what aspects of their games required improvement. In the Fall 2009 offering of the course, I made students give multiple prototype presentations. Once a week, each group gave a quick prototype demonstration in their lab or to the rest of the class during a lecture. During these prototype presentations, each group received feedback on how they could make improvements to their game from both their peers and the instructors.

Delivers high quality information to students about their learning. Although quality is defined quite broadly, ensuring that feedback is provided in a timely matter is certainly a significant part in the effect that the feedback will have on the student. Quality feedback must include some praise alongside constructive criticism, and is more precisely defined as information that helps students troubleshoot their own performance and self-correct (corrective advice). Thus, it is important that the students understand the feedback's relation to goals, standards, or criteria. Limiting the amount of feedback is also important, so students do not feel overwhelmed, and are able to actually use the feedback. The first assignment in our class is focused on how to give an effective review. Early in the semester, lecture and lab time was dedicated to instructing students on how to give effective *peer* review. Students presented their first game projects during a class lecture, and received feedback live by a panel of instructors. This provided the rest of the class with examples of the types of feedback we were expecting them to deliver in the subsequent peer review activities. I hoped that by doing this we were preparing students to enable them to be effective and confident when giving feedback to their peers throughout the rest of the semester. In the game prototype demonstrations, I limited the number of peer responses so groups were not flooded with suggestions. I also often split the audience into two sections, where one section was responsible for providing praise about what they really liked about the game prototype, and the other section suggested possible improvements to it. As these prototype demonstrations happened at each stage of the game development process, students were able to immediately apply changes to their projects based on the feedback received from their peers and instructors.

Encourages teacher and peer dialogue around learning. Students who have just learned something are often better able than teachers to explain it to their classmates in a language and in a way that is accessible. Peer discussion also exposes students to alternative perspectives on problems and to alternative tactics and strategies. Student discussion about feedback is very important, and a suggested activity is to ask students to find one or two examples of feedback that they found the most useful and explain how and why that particular feedback helped. There were two to three online weekly quizzes in our course, where a question was posted on the class forum, and each student had to post a unique answer to the problem with a resource they used when working through the problem. This allowed students to learn a multitude of possible solutions to a given problem, as well as view the resources different students found useful. To provide a unique answer, students were forced to check on solutions previously posted to the forum before posting their own solution. We also introduced some class activities where students had to evaluate their peer's solutions to previous quizzes. Students also actively used the forums to discuss and aid each other through assignment work, by posting helpful tutorials or helping other groups with suggestions. The game prototype presentations were also effective tools for peer learning, for in addition to receiving suggestions on how to improve work, the presenters were often asked how they implemented certain things their peers were impressed with. I used the labs as a place where students could discuss current assignments and quizzes, and often members from assorted project teams discussed the different ways their groups had implemented certain features in their games.

Encourages positive motivational beliefs and self-esteem. Research in school settings has shown that frequent high-stakes assessment has a negative impact on motivation for learning. Such assessments encourage students to focus on performance goals (passing a test) rather than learning goals (mastering the subject). Butler demonstrated that feedback comments alone increased students' subsequent interest in learning when compared with two other controlled situations, one where only marks were given, and the other where students were given feedback and marks [15]. Butler argued that students paid less attention to the comments when given marks, and consequently did not try to use the comments to make improvements. It is suggested that instructors provide comments on submitted work, and then give students a chance to respond to the feedback before the work is graded. Another option is to have students submit drafts of their work to be assessed before their final submission, and/or allow resubmissions. Often instructors do not have the time to mark submissions multiple times, and I believe the prototype presentations we introduced into our course are the perfect solution to this problem. The presenters are not being graded on their prototypes, but they are being provided with multiple sources of feedback, and integrating these activities into a course does not require a significant amount of extra time or effort from course instructors.

Provides opportunities to close the gap between current and desired performance. This shifts the focus away from simply the quality of feedback towards how that feedback leads to changes in student behaviour. As Boud notes [14]:

> "Unless students are able to use the feedback to produce improved work, through, for example, re-doing the same assignment, neither they nor those giving the feedback will know that is has been effective."

Boud argues that feedback should help students to recognize the next steps in learning and how to take them, both during production and in relation to the next assignment. It is suggested that students should be provided with feedback for work in progress and have increased opportunities for resubmission. Another option is to have two-stage assignments where feedback on stage one helps improve stage two. This principle is the essence of studio-based learning, which asserts that feedback should be given throughout the course of an assignment so that students can put the feedback to use. The first of our two game projects is worth much less than the final project, and we discuss with students that the first project is intended to be an introduction to working in teams, the game design process, and programming in Flash. We assert that they should apply what they learn from these experiences to their final projects. We also provide them with a lot of feedback during their first project presentations. The game prototype presentations also provide students with an opportunity to immediately apply the feedback they receive from their peers to their current project multiple times before they are graded on it.

Provides information to assessors that can be used to help shape the teaching. Good feedback practice is not only about providing usable information to the student being assessed, but it should also provide good information back to the assessors. The suggested ways of obtaining this information is by allowing students to request the feedback they would like on a project, or by allowing students to identify where they are having difficulties. During the prototype presentations, often students demonstrated a few possible ideas they were considering implementing in their game. They used the feedback received by their peers to help them make a decision on some of their difficult design choices. Some groups also used these prototype presentations as a time to showcase different features they felt were very hard to implement, effectively motivating class discussion on different strategies used to overcome certain obstacles found during game development. The assessors also often queried the presenters about the implementation of features the assessors were personally interested in. These examples show how these prototype presentations proved to be very effective learning tools for both the presenters and their reviewers.

3.2 Course Outline

In the following subsections, I first concretely identify common learning objectives shared between our CS-games and CS-traditional courses offered in the Fall 2008 semester. I then overview significant ways in which the courses differed in terms of design and delivery. Subsequent subsections will list the differences between the Fall 2008 and Fall 2009 games courses, and explain the reasons behind our course redesign decisions.

3.2.1 Learning Objectives and Assessment

The learning objectives were introduced and discussed in the lectures, and assignments and quiz activities were used as a tool for students to progress towards reaching certain learning outcomes. The learning objectives shared in both courses were the software development process, specifically design and documentation, evaluation, and prototyping. Rules, balance, algorithms, complexity, compilers, state machines, and binary logic were covered once the students were comfortable with the introductory material. Finally, the concepts of opponents, artificial intelligence, networking, interfaces, human-computer interaction, graphics, audio, and ethics were explored.

In total, there were 20 quizzes, four assignments, and one final project in the CS-games course, and five assignments, seven quizzes, and two mid-term examinations in the CS-traditional offering. More quizzes were created in the CS-games course, to cover the learning objectives that were not accounted for in the game-based assignments.

3.2.2 Assignments

The CS-traditional course had five assignments: building a web-page, writing algorithms in pseudo-code, using binary logic and circuits to represent solutions to problems, solving deadlocking problems, and creating a game in Game Maker. With the exception of the Game Maker assignment, in which students could form their own groups, all assignments in the CS-traditional course were completed individually.

The CS-games course had four assignments, and a final project. Students were placed pseudo-randomly into groups for assignments, and group sizes ranged from three to five members. In the first assignment, students had to extensively play-test a game, and then submit a written review. For the second assignment, students made an elevator pitch in the lab about a game concept of their own design. At the end of the lab, the students voted on their favourite pitches and were placed into groups to extend the pitch into a full design document. For the third assignment, we used Game Maker, and scrambled the students into new groups and had them implement a game chosen at random from the compilation of design documents submitted in the previous assignment. Our fourth assignment used Inform, and the students created a text-based adventure. For the final project, students again made elevator pitches to their peers, and then voted to determine top pitches. This time, however, teams went through each stage of the development process together, and could choose to build their final game using a language of their choice. The students had to present both games to the rest of the class, and were randomly assigned three text-based adventures to peer evaluate.

3.2.3 Quizzes, Labs, and Textbooks

For the CS-traditional course, two mid-term examinations, as well as quizzes given out every two to three weeks, covered the learning objectives not investigated in the assignments. There were 10 lab sessions, and an online textbook resource [39], used to support topics currently being explored in assignments.

Instead of having mid-terms or a final exam, the CS-games course had two to three weekly quizzes on topics covered in lectures. Quizzes were done in a WIKI-format, as the course stressed cooperative learning and peer review. In our WIKI-quizzes, students were able to see their peers' submissions, but had to submit a unique solution of their own. This also led us to the decision not to use a textbook, as the students naturally created an online collaborative resource on the topics covered in the course with their collective quiz submissions.

There were 12 lab sessions in the CS-games course. We designed the lab activities to provide the students with the industry experiences we wanted to address. All of the lab activities involved cooperative group work, and three of the labs had a major emphasis on peer review. We created games in Game Maker, Inform, and Future Pinball[3], so that students were introduced to pre-existing code bases in three labs. The focus of these labs was to familiarize the students with a program and then let them modify and add to it.

[3]http://www.futurepinball.com

3.2.4 Group Organization

Deibel argues that greater student interaction and learning can take place by using instructor-selected groups, and was successful in raising students opinions in regards to group work [21]. Deibel talks about the latent jigsaw method of forming groups, a new method for assigning groups based on prior student knowledge for the purpose of peer-teaching. In this method there is an expert in each group that teaches their expertise to the rest of their group. Begel's recent industry study recommended that if students could interact with a more experienced colleague, it would be a valuable experience [11]. Although the CS-games class was officially a first year course that did not require any prior programming knowledge, some students in the class had previous programming or even game design experience. After completing the first few weeks of lab activities it was apparent which students had programmed extensively before. Students were also very vocal about their own programming skill and experience (or lack thereof). From the students perspective, the groups were created randomly, as for each assignment they worked with a different selection of people. In truth, we tried to keep the experienced programmers apart, so that they could act as programming leaders in their selective groups. This approach ensured that teams were comprised of members with different skill sets, to the extent that was possible within our course. For the final project, we also wanted to make sure students were able to work on a project they were genuinely interested in, and so tried to group students that shared similar game genre interests.

3.3 Addressing Industry Issues

In a study on the experiences of new college graduates in their first development job, it was found that university Computer Science curricula provided the recent graduates with adequate design and development skills, but that their communication, collaboration, and orientation skills were not as well addressed [11]. Jackie Copland, from Electronic Arts (EA), provided a guest lecture for our students, and presented a video interviewing recent university graduates now working at EA, based on Colleen McCreary's "Living the Dream" talk [44]. The presentation helped demonstrate the significance of these non-technical issues to the students, and how important they are to the development process.

3.3.1 Cooperative learning

A common problem regarding communication was discussed in McCreary's video [44]. Programmers were generally unable to communicate well with members of other teams that did not have strong technical knowledge. In other first year Computer Science courses offered at the University of Victoria, programming assignments are done alone, and in group projects, all members usually have a similar level of programming experience and technical knowledge, or at least the same lower bound. I created groups for each of our assignments so that they were composed of students in various disciplines, with a wide range of technical knowledge.

In our WIKI-based quizzes, students could view their peers' submissions, but had to submit a unique solution of their own, along with a unique resource they used to help solve the problem. This exposed students to many different approaches to a given problem. Many quizzes were very open-ended, and I thought it was important for students to see how their peers from different disciplines or backgrounds interpreted questions differently, and how that interpretation affected their solution. Many of our students learned concepts in different ways, and the variety of different posted resources helped other students fully understand difficult concepts. I also set up group forums for each of the assignments on the Moodle web-page. This was so students could discuss their assignments, and get help from other members of their group on parts they were finding particularly difficult.

3.3.2 Peer Review

McCreary's video also talked about peer review [44]. It was noted that when new graduates come into the work environment they are paired with a *programming buddy*, and that there are frequent code reviews. EA finds that new workers get frustrated and upset during the peer review. They claim it would be much more helpful if students got used to having to show their work, and walking people through the decisions they have made and why they made them. After being provided with these types of experiences, when they get into the work environment they should be more comfortable explaining their decisions. The group forums on our Moodle site provided a medium for students to share with group members parts of their work, and explain their decisions. Other group members would essentially review the work and give their personal feedback.

Our WIKI-based quizzes also involved peer review. Students analyzed the submitted solutions before completing and submitting their own. The benefit of an early submission was the fact that many options were available without the risk of a similar solution. The trade-off was that students who submitted a quiz early, had to defend their solutions from the critiques of the rest of the class. For one of the daily quizzes, students had to write an algorithm, and the following day the students were assigned to evaluate and critique a submitted algorithm.

I used some of the peer review activities as part of the course assessment. For participation marks, I assigned every student three text-based adventure assignments, chosen pseudo-randomly, to review and evaluate. Also, when students presented their game created in Game Maker to the class, for participation marks, students in the class were required to ask two questions or give two critiques about other groups' games being presented.

After assignments had been submitted, I always had students evaluate group members on the following factors: reliability, communication, helpfulness, easiness to work with, and overall value to the team. Students had 5 points per question to distribute among their team members. This section was followed by a place where each student could make anonymous complimentary comments or constructive criticisms about each member of the group, and/or the actions by their peers they found to be the most helpful or unhelpful.

3.3.3 Orientation with a Pre-Existing Code Base

Also from the industry studies, it was reported new workers have trouble working with large code bases, when they have to work on a particular section of a large project. New workers typically find it difficult to efficiently create solutions that correlate with the pre-defined infrastructure. Egert states that often new developers join large, pre-existing teams of software developers as the most inexperienced member, and spend their first several months resolving bugs that predate their employment, with little access to easily consumable documentation [11]. As students work their way through college, they usually either build their own systems, or have the infrastructure explained in detail during lectures or labs. Egert also discusses how students try to do everything themselves, and try to fix every bug along the way, even if they do not have time.

The key goal of our group projects was to give students experience working with implementations unfamiliar to them. The groups should also help students become comfortable discussing implementations, as opposed to trying to complete more tasks than they realistically have time for.

As our course was not primarily a programming course, and there were students without any previous programming experience enrolled, it was difficult to incorporate activities involving a large pre-existing code base into our design. As a first step, I introduced our students to Game Maker, led them through a game I had created, and let them tweak different variables and functions and see the consequential result. I did a similar assignment with Inform, where students had to learn the rules of pre-existing infrastructure. Students then had to override certain commands that did not function as was required for their implementation, as well as add their own rules and commands to the pre-existing collection.

As a lab assignment, students created a pinball game using the Future Pinball program. The program uses a limited set of the Visual Basic Scripting Language. The main engine of the game is already programmed, but it is designed to be flexible enough to allow for a wide variety of games to be created. Through a tutorial, and trial and error, students learned how the pre-existing scripts functioned so they modify and add to the script to personalize their pinball games.

3.4 Changes made for the Fall 2009 offering

The content of the Fall 2009 course on game design was very similar to the previous offering of the course at UVic that we introduced in the Fall 2008 semester [26]. The main activities I restructured were those involving peer review; where possible, I preserved the other aspects of the course so that I could get an accurate assessment of how our refined peer review activities affected student engagement. The number of students rose from 87 to 115, so the class enrollment was again high enough to easily divide students into groups ranging from three to five members for each project. Again, I used Moodle as our course management system, as using the Moodle system provided us with an easy way to manage groups, and to create forums and online quizzes.

Students created three games in the Fall 2008 offering of the course, but after the course was over, students suggested decreasing the number of assignments, as they found it difficult to create three different games in three different groups within a 10 week period. I decided to look at prior work on project-based courses involving group work. McConnell states that when a group is formed, there is a portion of time where the group members are learning about each other, and in this time the group is less effective [43]. Over time, the group members start to understand how to best communicate with each other, and McConnell asserts that this process can take four to seven weeks depending on the frequency of group interactions. Richards completed a follow up study, and found similar results regarding time requirements, and also reported that students prefer to change groups between projects [55]. I opted to decrease the number of game projects in our course down to two, both ranging between four and seven weeks in length. We provided time in both our lectures and labs so groups were able to meet multiple times a week, enabling group members to sufficiently familiarize themselves with one another. Students created games from the ground up in each of the two game projects. For the Fall 2009 course offering, we chose to use Flash[4] as the tool to develop games, instead of Game Maker, which we used in the Fall 2008 offering. We initially chose Game Maker because it is widely considered very intuitive and fun, and therefore often used in introductory game programming courses. Since our course does not have a programming pre-requisite, we hoped drag-and-drop nature apparent in Game Maker would ease students into game programming. About half-way through the first game project in the 2008 offering of the course, most groups had already switched to Game Maker's built in scripting language, reporting that they actually felt restricted by Game Maker. Flash has a much steeper learning curve than Game Maker, as students use Actionscript to program their games in Flash, which is a coding language. Despite the increase in difficulty, Flash does give game artists much more flexibility and scalability, is very easy to put up on a web-page, and when put on a portfolio is much better suited to industry demand. The lab activities throughout the first few weeks of the semester were devoted to introducing students to Flash. Tutorials on animation and Actionscript 3 were given, so students were able to choose an art or programming role when placed into a group for the two game projects.

Before work began on the game projects, each student presented a game concept pitch to their peers in their weekly lab section. Students then voted on their favourite pitches, and the top-voted game pitches from each lab section were selected to be developed. Students were pseudo-randomly put into groups to work on each game; this year I made sure each group was constructed so that it consisted of at least one student willing to be a programmer and one willing to be an artist. Each group was required to produce a design document for a game, followed by the creation and then

[4]http://www.adobe.com/products/flash

demonstration of a working game to the rest of the class.

To gradually introduce students to the game development process, the first game project spanned four weeks, and was designed to be much smaller in scale than the final project, which was allocated six weeks. One of the main objectives of the first game project was to get students comfortable working in groups, using Flash, and acquainted with the game development process. I suggested simpler designs for the first game project, for in the Fall 2008 offering of the course we noticed students often attempted to create a game much more complex than was possible to complete given their time constraints. After students had finished their first project, and were better able to determine what was possible to accomplish in a six week period, I allowed students to set their own limitations to the scope of their game for their final project.

3.4.1 Peer Review Redone

When restructuring the peer review activities for our course, I kept in mind the significant issues apparent in other studies on peer review: issues such as students feeling uncomfortable with peer review activities, feeling unqualified to give effective feedback, and participating in conspiracy activities. Fortunately, previous studies also provided me with a lot of information about what types of peer review activities could potentially lead to positive results. Based on what I learned from previous studies, I decided the right approach was to begin with lectures on how to both effectively present ideas and constructively critique one's peers' ideas.

To prepare the students for the peer review activities, for the first assignment, students assumed the role of a typical game reviewer. I hoped external reviews would help students feel more comfortable acting as a peer reviewer in future activities. In two weekly assignments given at the start of the semester, students were required to review games developed outside the classroom. The first review was of a previously released console or computer game, in which the students were asked to write a review similar to one they would find in a typical game magazine or Web site. In the second review activity, students gave feedback on a game being developed in Victoria, placing students into a game tester role to give students experience in providing constructive criticism. They received feedback from the course instructors on how effective their feedback was, further readying them for later peer review activities.

Peer review activities were gradually introduced into the course, starting with a point-counterpoint activity. The students were given a topic and asked to post to the forum two reasons why they were in favour of it. They were then asked to refute the points brought up by another student. Our first point-counterpoint activity was intended to be a lightweight and fun topic, specifically, why Batman was greater than Superman. The objective of this activity was simply to get students comfortable commenting on one of their peer's work. Student presentations were also introduced into the course as smoothly as possible. Students made one to three minute game pitch presentations for their first game assignment to the rest of their lab, instead of to the whole class, where lab sizes ranged between 15 and 18 students. Although feedback was given to the presenter, the presentation was graded only for participation, as I wanted to give students a chance to give a presentation before being evaluated on their presentation performance.

Within their assigned game groups, students presented the finished versions of their first games to the whole class. Although other students were welcome to comment on the game presentation, the core feedback, as well as the grade, was administered live by the course instructors. Three judges sat at the front of the audience and commented on the work after each group presentation had ended. Feedback was given primarily by the instructors to give the class examples of what type of feedback was expected from them in later peer review activities.

A studio-based model was adopted for the second game project by integrating peer review activities into every step of the game development process. Each team had to give multiple presentations, in both their lab sections and to the rest of the class, while progressing through their final projects. The first presentation was the initial game concept, followed by a few game prototype demonstrations, and then a practice final presentation a few days before the final game project deadline. The student feedback for each presentation was marked only for participation, as was recommended by previous studies.

Table 3.1: Project milestones involving peer review

Activity	**Project 1**	**Project 2**
Game concept	✗	✓
Game prototypes (3)	✗	✓
Practice presentation	✗	✓

On the class forums, I posted some of the comments I felt were very helpful, and asked each student to post which comment they personally felt helped him or her the most. Students gave quick demonstrations, similar to progress reports, during some of the lab sessions, and the remainder of those lab sessions was dedicated to peer and teacher discussions about possible improvements to different aspects of the games. During one of the prototype presentations I split the class into two groups, where one half was responsible for commenting on positive aspects of the prototype, and the other half responsible for constructive critique. In the practice final presentation, students were asked to peer review other groups' games in a similar manner to how the instructors had previously judged and graded the first game presentations. Table 3.1 shows how I integrated peer review activities into the milestones of the game development process in the second project. I discuss our observations on the effectiveness of the studio-based model into our final game project in section 6.

Chapter 4

Analysis: Fall 2008 Offering

The following subsections overview an analysis we performed on our CS-games and CS-traditional courses. These courses ran for 13 weeks in the Fall of 2008, and students were free to select between the two course options. Enrollment in CS-games was 87, whereas enrollment for CS-traditional was 38. Quantitatively, we present survey results and engagement rates. Qualitatively, we further explore our success with respect to our three key industry issues: collaborative practices, peer assessment, and code orientation.

4.1 Quantitative Results

Many of the students in both classes, CS-games and CS-traditional, filled out two surveys about their interest in Computer Science and game design. The first survey was given out at the beginning of the semester, and the second survey was given out after the students had completed all of their course work. In the 5 questions we are presenting, the students rated their interest using a 5 point Likert scale: 1 indicated no interest, 3 neutral, and 5 meant very interested.

In the first survey given in the CS-traditional class, of the 25 students who completed the survey, 84% of them listed themselves as Computer Science majors, and 92% of them stated that they had taken a university-level Computer Science course before. Of the 81 students who completed the first survey in CS-games course, only 49% of them listed themselves as Computer Science majors, and 60% of them listed CS-games as the first Computer Science course they had taken. As we expected, these numbers indicate that the gaming angle can indeed have a positive effect on enrollment, and particularly on students pursuing a major outside of Computer Science.

The first question asked the students to rate their interest in pursuing a degree in Computer Science, or a related field (such as Software Engineering). Table 4.1 shows that students in both classes were interested in a Computer Science degree: CS-games average for the survey given at the start of the semester was 3.94, and at the end of the course, the average was 3.79. The CS-traditional start of semester average was 4.28, and the final average was 4.27.

Table 4.1: Interest in pursuing a degree in CS

	CS-games		**CS-traditional**	
Rating	**Start**	**End**	**Start**	**End**
Very interested	42/81 (52%)	32/63 (51%)	13/25 (52%)	10/18 (56%)
Interested	14/81 (17%)	9/63 (14%)	8/25 (32%)	6/18 (33%)
Neutral	9/81 (11%)	7/63 (11%)	3/25 (12%)	0/18 (0%)
Not very interested	10/81 (12%)	7/63 (11%)	0/25 (0%)	1/18 (6%)
No interest	6/81 (7%)	8/63 (13%)	1/25 (4%)	1/18 (6%)

The second question asked the students to rate the likelihood that they would enroll in additional Computer Science courses in the future. The results in Table 4.2 show the CS-games average in the first survey was 4.41, and 4.35 in the second. The CS-traditional course first survey average was 4.65, and the second average dropped slightly to 4.50.

Table 4.2: Likelihood to enroll in another CS course

	CS-games		**CS-traditional**	
Rating	**Start**	**End**	**Start**	**End**
Very interested	53/81 (65%)	43/63 68%)	21/25 (84%)	15/18 (83%)
Interested	15/81 (18%)	9/63 (14%)	2/25 (8%)	1/18 (6%)
Neutral	9/81 (11%)	5/63 (8%)	0/25 (0%)	0/18 (0%)
Not very interested	1/81 (1%)	2/63 (3%)	1/25 (4%)	0/18 (0%)
No interest	3/81 (4%)	4/63 (6%)	1/25 (4%)	2/18 (11%)

The third question in the survey asked the students to rate their interest level in further developing their programming skills, as shown in Table 4.3. The CS-games overall average in the first survey was 4.39, and the average dropped to 4.15 in the second. The first average in the CS-traditional course was 4.50, and the second average was 4.33.

Table 4.3: Interest in further developing programming skills

	CS-games		**CS-traditional**	
Rating	**Start**	**End**	**Start**	**End**
Very interested	53/81 (65%)	35/63 (56%)	19/25 (76%)	12/18 (67%)
Interested	14/81 (17%)	13/63 (21%)	1/25 (4%)	4/18 (22%)
Neutral	7/81 (9%)	8/63 (13%)	2/25 (8%)	0/18 (0%)
Not very interested	3/81 (4%)	4/63 (6%)	1/25 (4%)	0/18 (0%)
No interest	3/81 (4%)	3/63 (5%)	1/25 (4%)	2/18 (11%)

The results from the fourth question, shown in Table 4.4, show how interested the students were in game design. In the first survey for the CS-games course, this question received the highest overall average, at 4.54, and dropped to 4.21. In the CS-traditional course, the first average was 3.92, with a final average of 3.42.

Table 4.4: Interest in learning about game design

	CS-games		**CS-traditional**	
Rating	**Start**	**End**	**Start**	**End**
Very interested	53/81 (65%)	32/63 (51%)	8/25 (32%)	5/18 (28%)
Interested	21/81 (26%)	17/63 (27%)	9/25 (36%)	6/18 (33%)
Neutral	5/81 (6%)	10/63 (16%)	5/25 (20%)	2/18 (11%)
Not very interested	2/81 (2%)	3/63 (5%)	1/25 (4%)	4/18 (22%)
No interest	0/81 (0%)	1/63 (2%)	1/25 (4%)	2/18 (11%)

Table 4.5 shows the final question on our survey, where we asked the students how interested they were in taking a more advanced course on game design. In the CS-games course, the first survey average was 4.40, and the by the end of the course offering it was 4.14. For the CS-traditional course, the first survey average was 3.82, and the second was 3.32.

Table 4.5: Interest in a more advanced course on game design

	CS-games		**CS-traditional**	
Rating	**Start**	**End**	**Start**	**End**
Very interested	51/81 (63%)	34/63 (54%)	9/25 (36%)	4/18 (22%)
Interested	20/81 (26%)	16/63 (25%)	5/25 (20%)	7/18 39(%)
Neutral	3/81 (4%)	5/63 (8%)	6/25 (24%)	2/18 (11%)
Not very interested	5/81 (6%)	4/63 (6%)	2/25 (8%)	3/18 (17%)
No interest	2/81 (2%)	4/63 (6%)	1/25 (4%)	3/18 (17%)

4.1.1 Survey Significance

The results from the first survey show that students who enrolled in the CS-games course generally had a high interest in both Computer Science and game design. Although the average interest levels dropped in every category, with such high levels of interest, this is what we expected as workload pressures set in. In the final survey, the overall averages were still high, as the overall decline in interest levels were all under 8%. In each of the 5 questions on both surveys, our results show that over 50% of the students rated their interest level as ‘very interested’.

These results suggest that in addition to increasing enrollment, students who completed the CS-games course still have a high interest in furthering their studies in Computer Science and game design. As our course was offered to any student at the University of Victoria, it is important to take into consideration the fact there are other causes for some of these effects. Other courses, part-time jobs, or personal factors that our survey did not take into consideration also may have affected our results. In future surveys, we plan to more thoroughly explore more causal indicators associated with these results.

In the following section, we will drill deeper into the results from the CS-games course to explore the issues associated with non CS-major’s interests.

4.1.2 Comparison: Games vs. Traditional

Overall, the interest changes in Computer Science topics between the two surveys in the CS-games course were more significant than those in the CS-traditional course. As the CS-traditional course was made up of a high percentage of Computer Science students, we separated the CS-games survey results into two groups. The first group was composed of the completed surveys where students indicated they were Computer Science majors, and the second group of surveys was made up of the students pursuing a different major. After the results were split up into the two groups, it was determined from an ANOVA test that the only case where a significant difference was observed ($p < 0.05$) was in the non CS-majors group. Most of the interest rating did not change in the surveys completed by the Computer Science majors in CS-games, and the results of this group more closely resembled the results of the CS-traditional course's surveys.

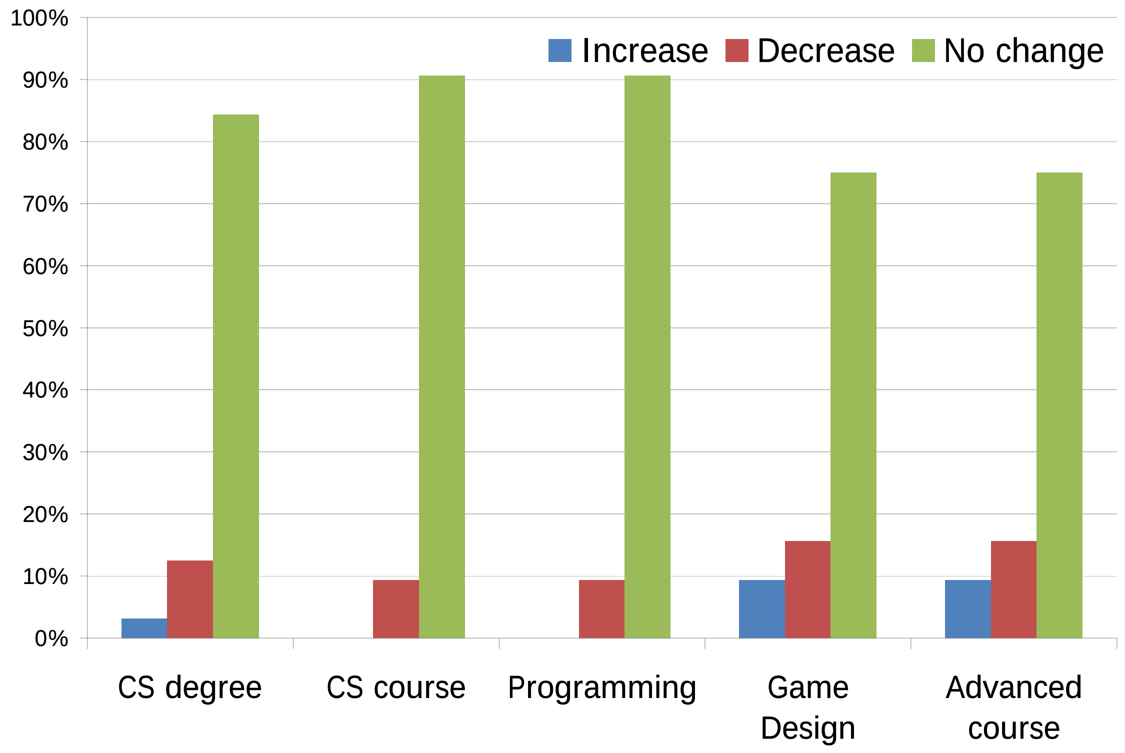

Figure 4.1: Change of interest for CS majors in CS-games (32 students)

Figure 4.1 shows changes in interest for Computer Science majors. The CS-games course offering did not have a significant effect on degree, future course enrollment, programming skill, game design, or advanced game course interest levels. Similarly, Figure 4.2 illustrates that there was no interest change for the majority of students in the CS-traditional course in Computer Science topics.

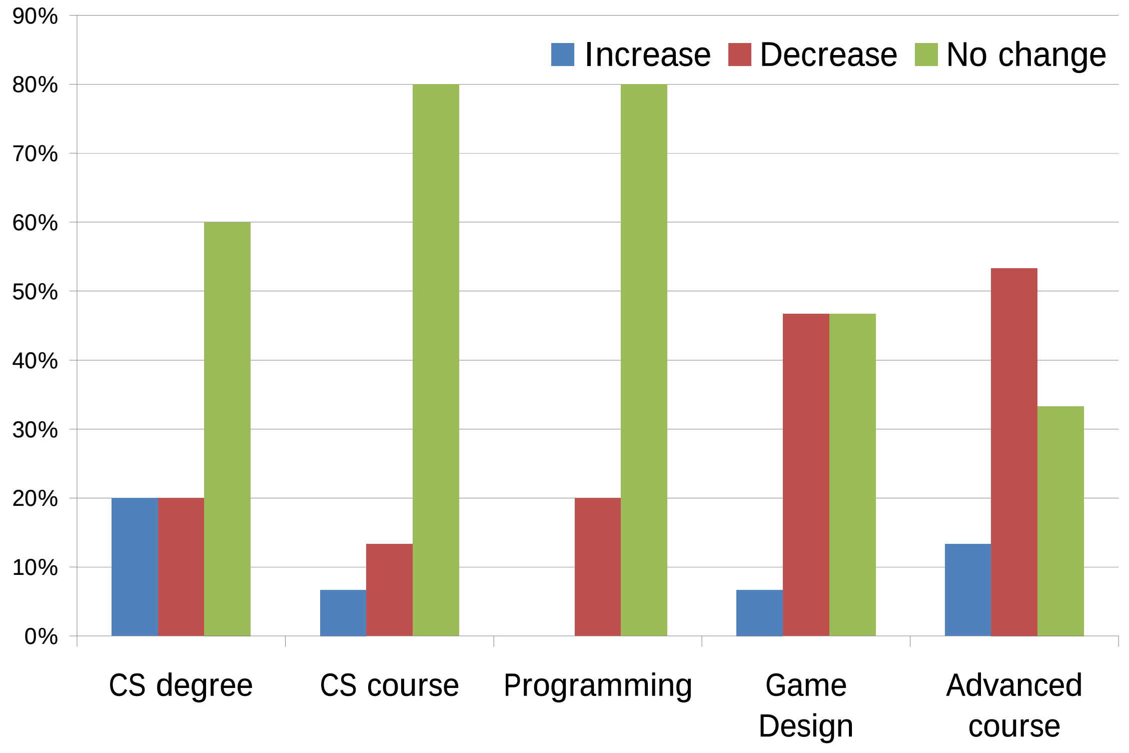

Figure 4.2: Change of interest for CS majors in CS-traditional (18 students)

The non-Computer Science majors in the CS-games course had much more volatile interest levels, as shown in Figure 4.3. A total of 31% of the students in this group had an increase in interest of obtaining a Computer Science degree, compared to the 23% whose interest decreased. Also, 31% of the students increased their interest in enrolling in another Computer Science course in the future, versus 19% whose interest level decreased. It is interesting to note that despite these positive results, the interest level of further developing programming skills decreased significantly, at 40%. This may be due to the fact that our course gave an overview of Computer Science topics, and focused on the design and development process of games, as opposed to a strong focus on programming. There was also a statistically significant decrease in interest about learning about game design, and a decrease in interest in taking a more advanced course on game design.

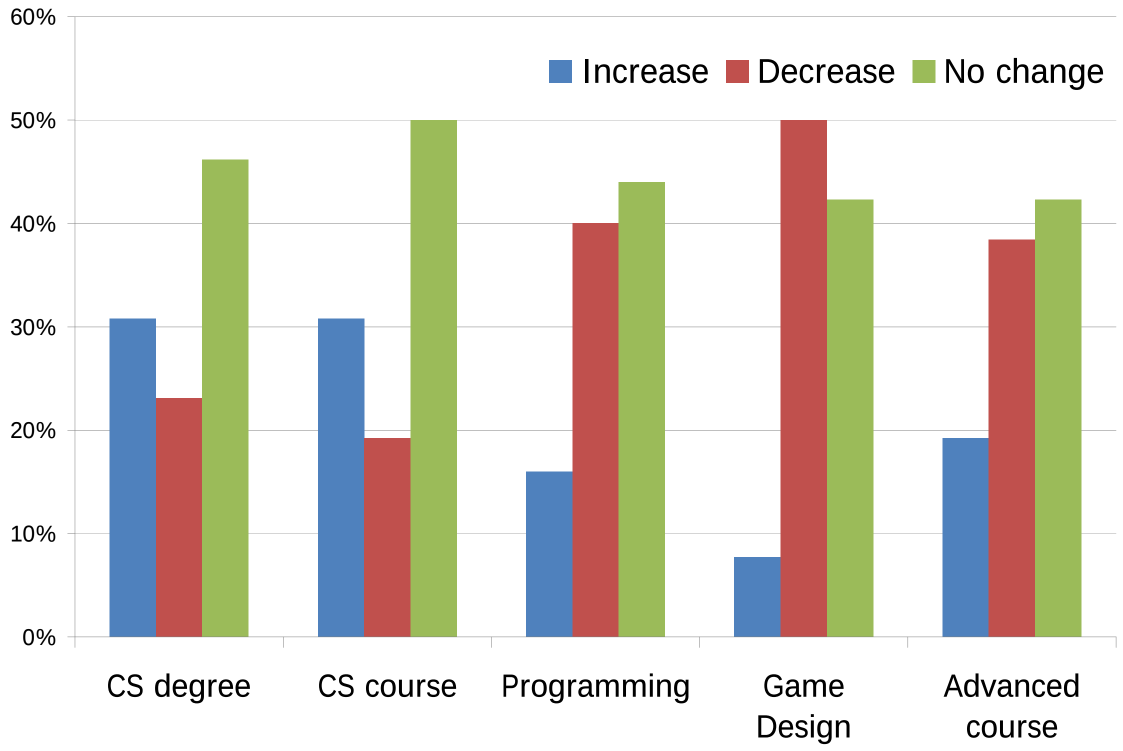

Figure 4.3: Change of interest for non-CS majors in CS-games (26 students)

Although the non-majors interests regarding a Computer Science degree and future enrollment are encouraging, the fact that there was a significant decrease in programming interest possibly indicates a flaw in our design. Perhaps because we stressed the non-technical issues present in industry, non-Computer Science students may have been given the impression that technical abilities are not key or necessary in obtaining a Computer Science degree.

4.2 Qualitative Results

In the peer evaluations students filled out after the assignments in the CS-games course, I added a section at the bottom for the students to give personal feedback about the assignment. I received a lot of mixed feedback about doing assignments in groups, as well as the software decisions we made for our course. In this section I overview feedback from the students in order to better qualitatively address the industry issues we attempted to target.

4.2.1 Cooperative Learning

The students generally liked the fact that we used Moodle as our course management tool. They really enjoyed using the forums and WIKIs, and often asked for extra forums to be created to discuss other relevant topics. Within the forums I created for the assignments, students often created new threads so other groups could play-test their games, to share useful tutorials, or to discuss how others implemented certain difficult mechanics in a game. The only qualm students had about Moodle was when they selected to be notified of forum posts and their E-mail inbox was filled with hundreds of notifications a day.

Some of the students that entered the course without any programming knowledge or experience commented that although Game Maker's drag-and-drop buttons were very intuitive, they felt that the drag-and-drop functionality was significantly inferior because they weren't progressing as quickly, and in some cases as effectively, as more experienced programmers. Some of these students felt their submitted assignment didn't reflect how much effort they had put in. One student E-mailed us about the problem:

> "Despite the simplicity of Game Maker, I had a hard time understanding it. I tried to get the game going from the time the assignment was up, while I was working on implementing features, such as the ammo/health display and cutscenes, I either ran out of time (not just the due date but deadlines set by others) or another group member did my job better than me. As such, none of my own work ended up in the final project."

Situations such as this discouraged some students from assuming a programming role in future assignments.

4.2.2 Peer Review

Students took the peer review activities very constructively and seriously. In a quiz about Game History, a student expresses how concerned he is about other students detecting a mistake:

> "Earlier today I posted that Doom was the top game in 1994, when that should actually have been 1993. Im not sure why I typed that. If you could edit my post to say 1993 so I don't look silly, that would be great. Thanks"

There was quite a difference in the quality of the game submissions for the third assignment. To acknowledge the groups that had worked extremely hard, I decided to have each group present their game to the class. We were astonished that 11 of the 24 groups asked if they could polish up their game for the presentation. Although they would not receive any extra marks for their updated version, most groups desperately wanted to present an impressive game to the class. I found it very interesting how students were more motivated to create a flawless product to show off to their peers, as opposed to the version submitted for marks.

Ultimately, 9 of the 21 groups ended up using Game Maker for their final projects. Students all wanted to impress their peers with their games, and became very excited as the presentation date approached. I received e-mails regarding Moodle privacy settings, as some groups were worried their secret presentation ideas would be stolen. Some groups got friends, family, and fellow University of Victoria students to beta test their games. The students' enthusiasm was apparent in the final presentations, as some groups presented trailers for their games, others used props like flashlights and expressive narrations.

4.2.3 Orientation with Pre-Existing Code Base

Some students without previous programming experience disliked the lab activities that involved investigating a pre-existing program. They felt they were too inexperienced to test and modify a program. One student stated that although he was very interested in the software process, he just could not code, and joined the course in hopes he could fulfill a different role in the game development process.

Although I did spend some lab sessions going over the basics of Game Maker, I completed other activities in the lab prior to the design document assignment.

Because of this, students made their first game pitches, and started their design documents, without being fully aware of the limitations of Game Maker. They were generally overly ambitious with their designs, and some students became frustrated with Game Maker when they were unable to smoothly implement their designs.

Chapter 5

Analysis: Fall 2009 Offering

The following subsections overview an analysis we performed on our game design course that ran for 13 weeks in the Fall of 2009 at the University of Victoria. Quantitatively, we present survey results from a survey distributed to the class at the end of the semester on how well the peer review activities were received, and how they affected the students with regards to course work. Qualitatively, we further explore the success of these activities through student feedback received during and after the activities.

5.1 Quantitative Results

Surveys were given out at the end of the semester, and 87 of the 115 students enrolled in the course fully completed the survey. Students rated how they agreed with each statement in the survey using a 5 point Likert scale: 1 indicated strongly disagree, 3 neutral, and 5 strongly agree. We used a binomial test, with a p-value of 0.05, when determining if results were statistically significant. Likert items were presented in a random order, however, we did not control for acquiescence bias. The following pages show the questions presented to the students in the survey, and the student responses. In the following section I discuss possible reasons for each survey question result by discussing what I observed throughout the Fall 2009 offering of the games course.

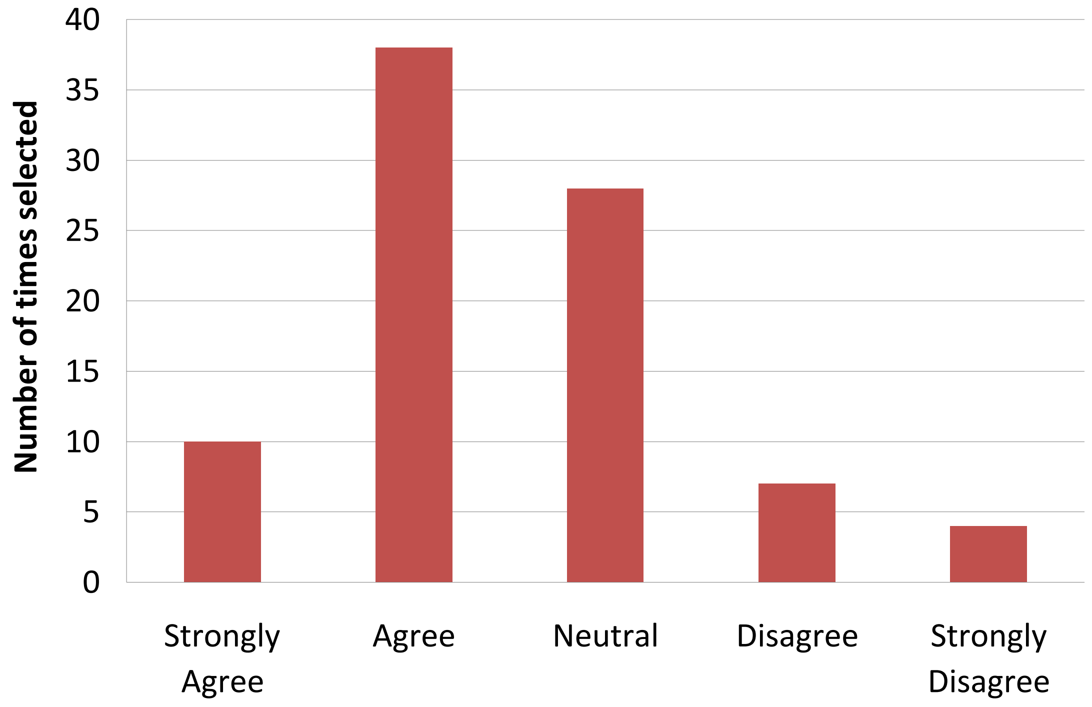

Figure 5.1: Survey results for the statement: "I feel this class should have had more peer evaluation." Average response: 3.49. Statistically insignificant (p-value = 0.1956).

I feel this class should have had more peer evaluation. The average response to this statement was 3.49, and this result is statistically insignificant. Strongly disagree was selected 4 times in this question, the most out of any question.

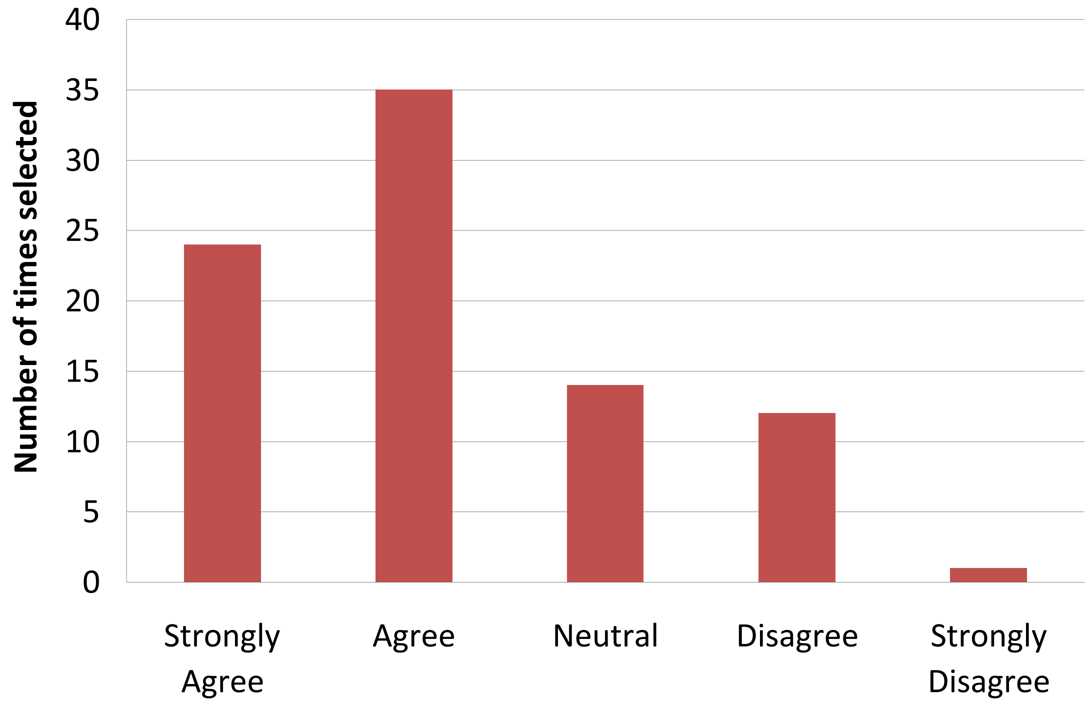

Figure 5.2: Survey results for the statement: "Giving and reviewing presentations to the class heightened my sense of competition." Average response: 3.75. Statistically significant (p-value < 0.001).

Giving and reviewing presentations to the class heightened my sense of competition. The survey results for this statement are shown in Figure 5.2, the average response was 3.75, and this result is statistically significant. The average response for this question was very high, and strongly agree was chosen as a response 24 times, the second highest number of times out of all of the questions.

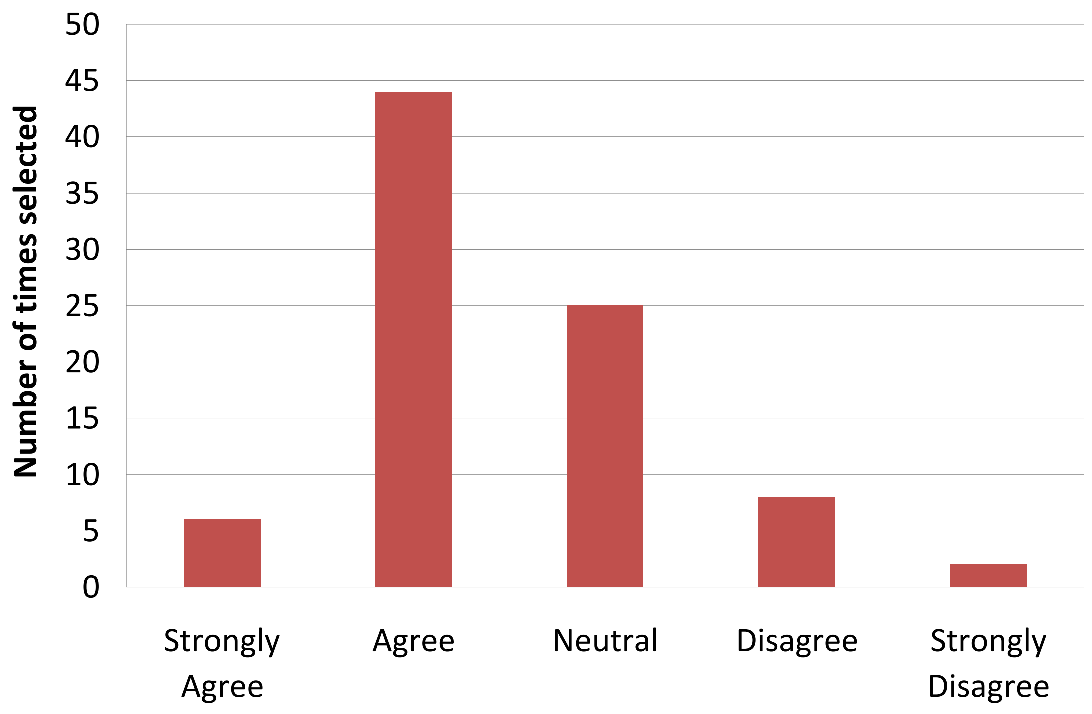

Figure 5.3: Survey results for the statement: “Peer Evaluation motivated me to participate more in class discussion.” Average response: 3.43. Statistically insignificant (p-value = 0.06).

Peer Evaluation motivated me to participate more in class discussion. The average response was 3.43, and this result was borderline (p = 0.06), but statistically insignificant.

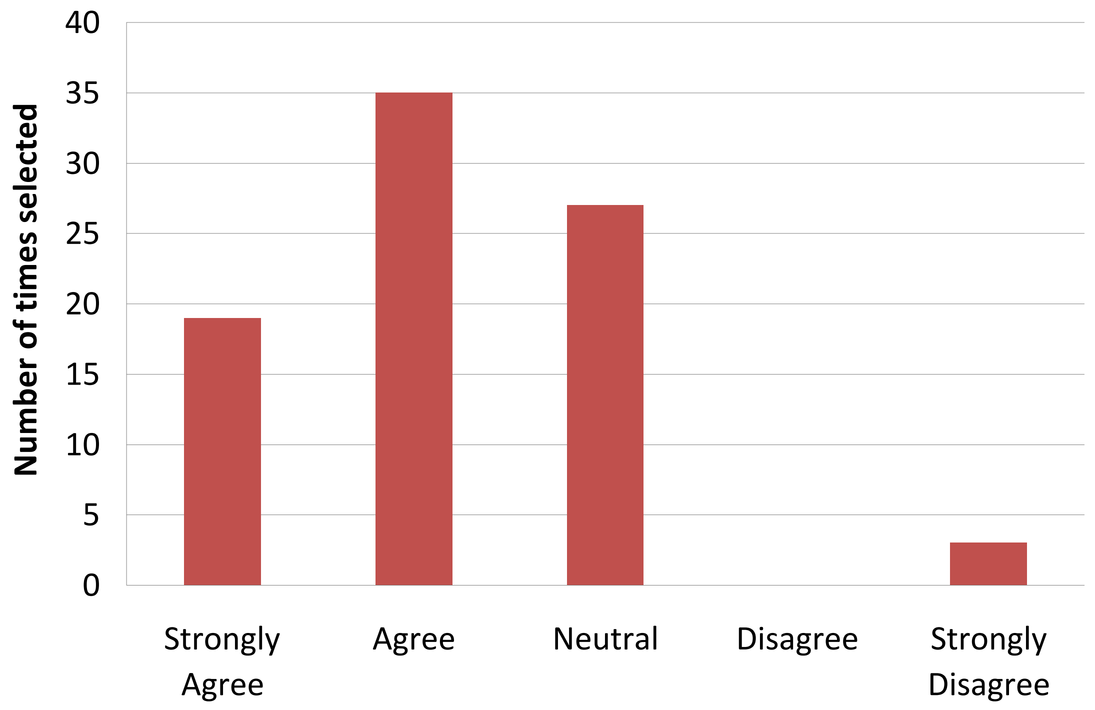

Figure 5.4: Survey results for the statement: "Giving and viewing class game presentations got me more excited about my coursework." Average response: 3.67. Statistically significant (p-value = 0.005).

Giving and viewing class game presentations got me more excited about my coursework. The average response for this statement was 3.67, and this result is statistically significant.

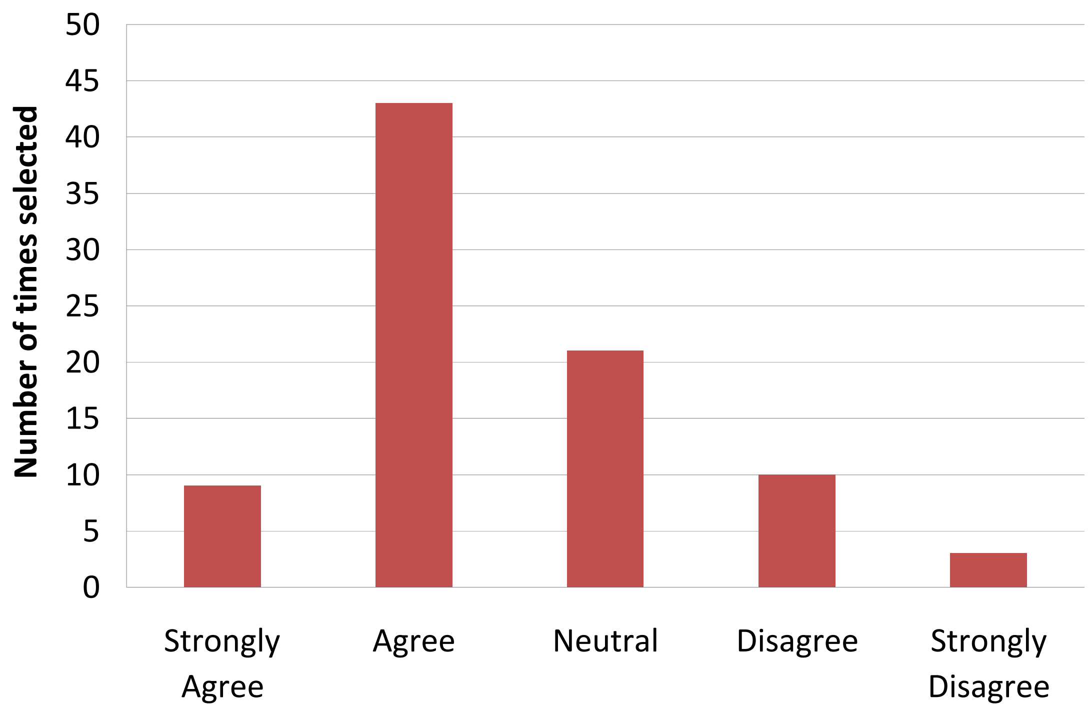

Figure 5.5: Survey results for the statement "I think my first game could have been improved if I had done a peer review session in the class or lab." Average response: 3.48. Statistically significant (p-value = 0.03).

I think my first game could have been improved if I had done a peer review session in the class or lab The average response for this statement was 3.48. Although the average is quite low, the result is statistically significant.

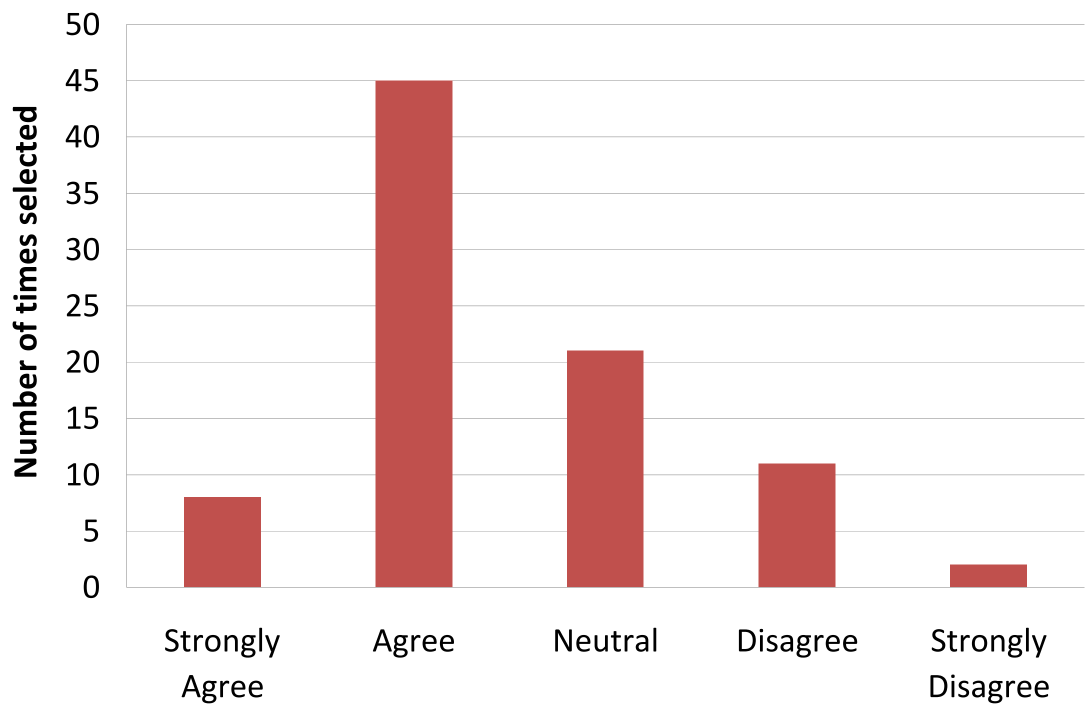

Figure 5.6: Survey results for the statement: "Peer review helped develop my sense of community within the course." Average response: 3.53. Statistically significant (p-value = 0.02).

Peer review helped develop my sense of community within the course The survey results for this statement are shown in Figure 5.6, the average response for this statement was 3.53, and this result is statistically significant.

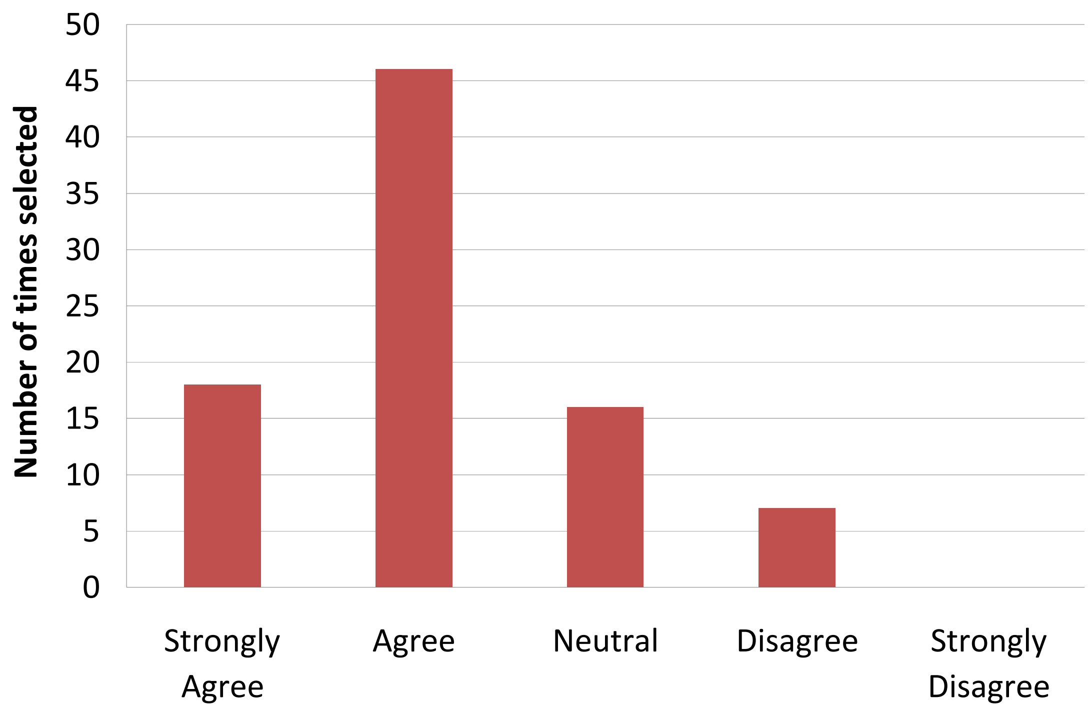

Figure 5.7: Survey results for the statement "Critiques and suggestions brought up issues/ideas I may not have considered myself." Average response: 3.86. Statistically significant (p-value < 0.001).

Critiques and suggestions brought up issues/ideas I may not have considered myself The survey results for this statement are shown in Figure 5.7, the average response for this statement was 3.86, and this result is statistically significant. This statement was tied for the second highest overall response average.

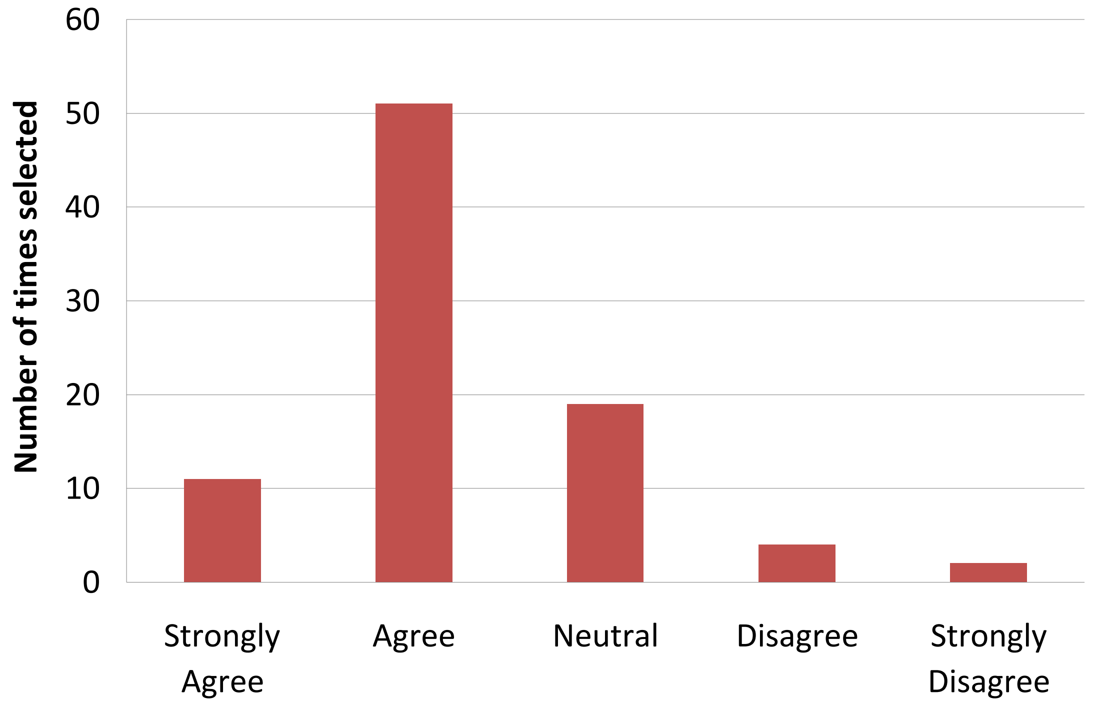

Figure 5.8: Survey results for the statement "After giving a few preliminary presentations, and receiving a round of feedback, I feel more confident presenting my game in the final presentation." Average response: 3.75. Statistically significant (p-value < 0.001).

After giving a few preliminary presentations, and receiving a round of feedback, I feel more confident presenting my game in the final presentation The survey results are shown in Figure 5.8, the average response for this statement was 3.75, and this result is statistically significant.

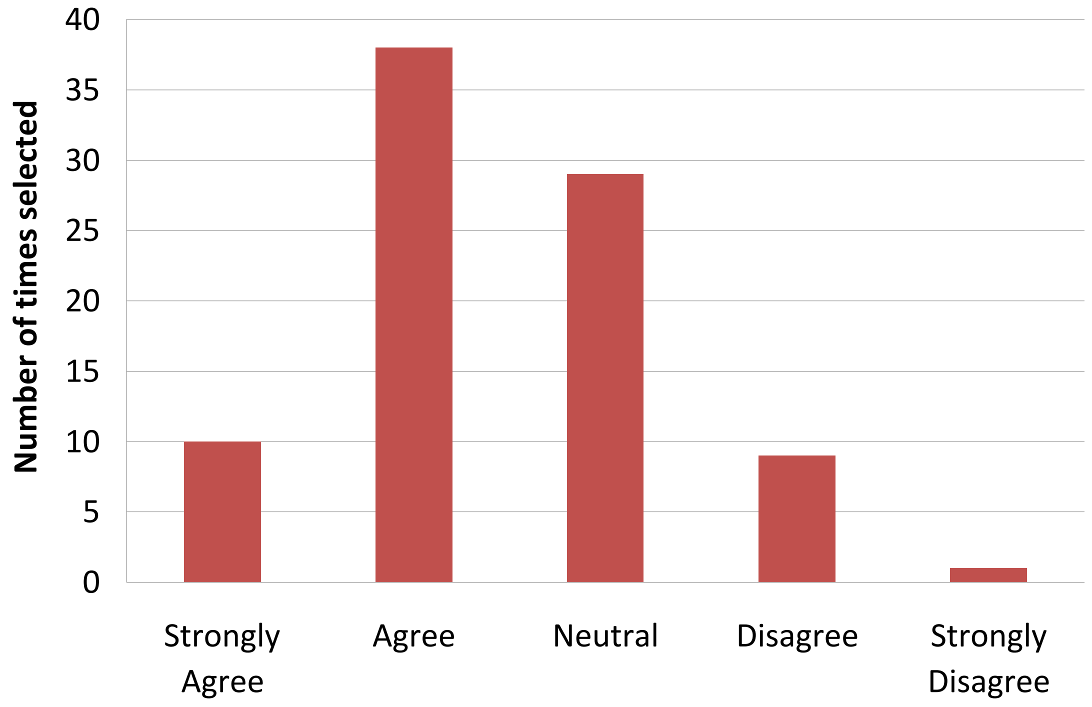

Figure 5.9: Survey results for the statement "This is a 2-part question, but the questions are scrambled. We did peer evaluation where students were asked to evaluate the second games, after seeing how the judges reviewed in the first games. I feel like after seeing what things the judges commented on I am NOW able to give proper feedback." Average response: 3.54. Statistically significant (p-value = 0.1956).

This is a 2-part question, but the questions are scrambled. We did peer evaluation where students were asked to evaluate the second games, after seeing how the judges reviewed in the first games. I feel like after seeing what things the judges commented on I am NOW able to give proper feedback. The average response for this statement was 3.54, and this result is statistically insignificant.

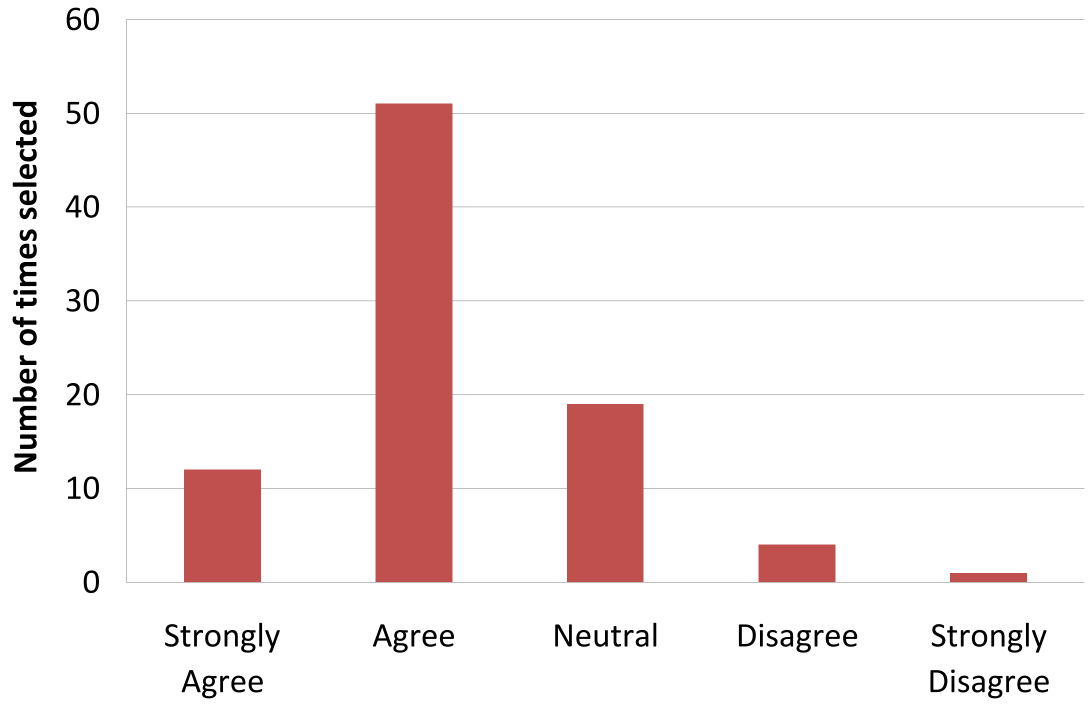

Figure 5.10: Survey results for the statement " This is a 2-part question, but the questions are scrambled. Assume we had done peer evaluation where students were asked to evaluate the first games, before the judges gave their thoughts. I feel like I would have been capable to give proper feedback." Average response: 3.79. Statistically significant (p-value < 0.001).

This is a 2-part question, but the questions are scrambled. Assume we had done peer evaluation where students were asked to evaluate the first games, before the judges gave their thoughts. I feel like I would have been capable to give proper feedback. The average response for this statement was 3.79, and this result is statistically significant.

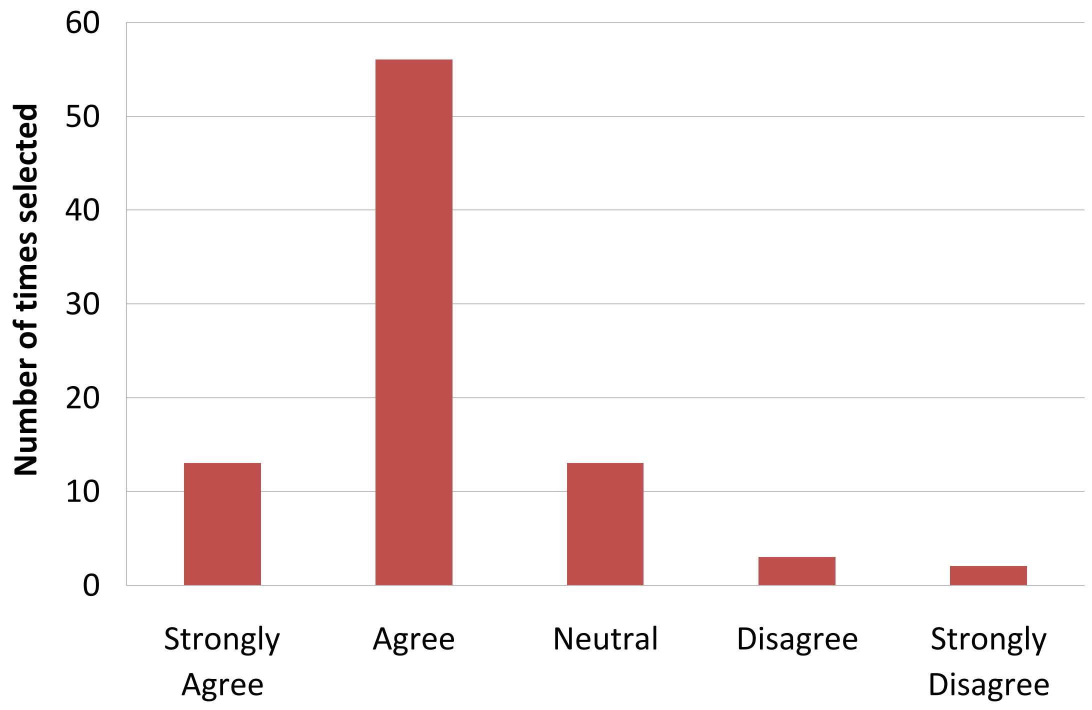

Figure 5.11: Survey results for the statement "Receiving constructive criticism and suggestions during the presentations motivated me to do better on my game." Average response: 3.86. Statistically significant (p-value < 0.001).

Receiving constructive criticism and suggestions during the presentations motivated me to do better on my game. The survey results for this statement are shown in Figure 5.11, the average response for this statement was 3.86, and this result is statistically significant.

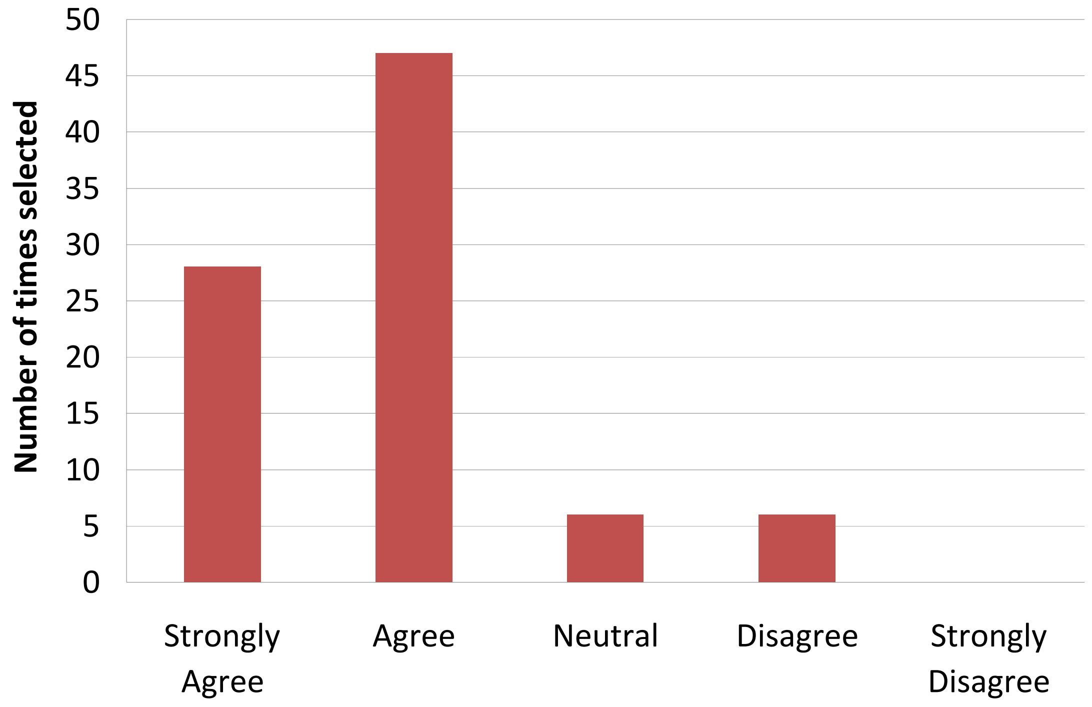

Figure 5.12: Survey results for the statement "Giving and witnessing other student suggestions to other groups' presentations got me thinking more about my own game." Average response: 4.11. Statistically significant (p-value < 0.001).

Giving and witnessing other student suggestions to other groups' presentations got me thinking more about my own game. The survey results for this statement are shown in Figure 5.12, the average response for this statement was 4.11, and this result is statistically significant. With an average of 4.11, this statement is definitely the strongest finding.

5.1.1 Survey significance

The surveys given out in the Fall 2009 offering were given out during the final week of the semester. Students had already completed all of their course work except for their final game presentations. As I did not give out surveys at both the beginning and end of the semester like we did in the Fall 2008 offering, I was not able to compare how our activities affected students interest in computer science. The viability of pre- and post-surveys in a first year course is often questioned, as first year students often have misconceptions about what they expect in a university course, and as they become accustomed to university life their opinions change, often greatly skewing survey results.

In the Fall 2008 semester, we ran the games course at the same time as we run the traditional computer science course, and so we had a control group with which to compare the success of our course. Unfortunately for the 2009 offering, the survey was only distributed to the games course, so I am unable to compare the results with a more traditional first year programming course. Setting up a control group, whether it is in another class, or by dividing students up into different control groups within the same course, may better evaluate the effectiveness of our activities. Control groups are often very difficult to set up; in the Fall 2009 offering there were a number of students that had previous experience participating in peer review activities, and were quite comfortable both giving and receiving critique. Students with prior knowledge or experience could greatly skew the results of a study, which was one of the reasons we were not able to create control groups within our course.

There were no questions regarding programming, or other computer science concepts we taught in the course. As this course was offered as a first year elective, there were students with fairly extensive programming and even game development experience. Although many of the students did learn about many computer science and software engineering concepts during the lectures of our course, the lessons would have merely been a review for senior level students. Thus, we did not place questions on the survey about how well students learned different concepts in our course, but structured our questions in an attempt to measure how effective students felt our activities were in increasing their levels of excitement and motivation, as well as boosting their presentation and peer review confidence. I was also interested in how valuable the students felt our new peer review activities were in helping them reach their learning goals.

5.2 Qualitative Results

Similar to the Fall 2008 offering of our games course, in the Fall 2009 offering I encouraged student feedback during and after all of our course activities. As expected, there were some complaints about group cohesion, but for the most part I perceived that the group work went smoothly. I received positive feedback about using Moodle, and overall it seemed that students enjoyed the switch from Game Maker to Flash. Game Maker's drag and drop functionality is very appealing to inexperienced programmers, in comparison to using Actionscript in Flash. The Actionscript syntax was intimidating for students that had never programmed before, and there were some complaints about how students had heard that in the previous year students got to work with Game Maker, in which "programming was not necessary". After the introductory tutorials on art and animation, all feedback from the artists in groups was in favour of Flash. During a class discussion, three student artists reported that they had bought Tablets to more efficiently create artwork for their teams. On the class forum, students posted useful Flash tutorial links, which were very beneficial for students new to programming. By the end of the first game project, it seemed that even the programmers warmed up to Flash, as for the final game project, we let students choose which environment to create their game in, and 26 of the 28 groups chose Flash. In comparison, for the final project in the Fall 2008 offering, 9 of the 21 groups chose to work in Game Maker.

I perceived that the students valued and enjoyed the peer review activities. Although a few students asked if the first presentation was mandatory, it was only the first presentation that students voiced any qualms about; students actually seemed increasingly enthusiastic about giving presentations as the course progressed. Although the survey results showed that students did not feel like more peer review activities were necessary, some students did 'extra' peer review. Two of the prototype presentations performed by the students were done in the labs, which ranges from 12 to 16 students. Some students attended multiple labs, so labs in addition to the one they were enrolled in, to receive feedback from students in other lab sections that they would not normally receive feedback from. This also provided students in the other labs with extra feedback during their own presentations.

The peer review activities also resulted in some interesting feedback from the students. A number of groups asked if they were able to hide certain features of their game during the prototype presentations, as they were worried some of their novel ideas would be stolen by other teams. Other groups used the peer review activities as a means of demonstrating possible new ideas and different possible directions for their game. They used the feedback they received from their peers to guide them towards some important design decisions. On the forums, we noted that comments about usability were commonly what students felt were the most important or helpful. Many groups reported that they had overlooked making game objectives and controls clear and intuitive for a new player.

Chapter 6

Observations and Discussion

6.1 Fall 2008 offering

Similar to the results found in previous studies, here at the University of Victoria, we also found that a game-focused course does increase enrollment. More than twice as many students enrolled in the CS-games course (85 students), as compared to CS-traditional (28 students). There were even two students who completed all of the assignments in groups without officially enrolling into the CS-games class (due to size limitations). One claimed that although he could not enroll in the class, he wanted to participate in the assignments for the experience, and because the course assignments were "fun".

Students were very enthusiastic throughout the first two assignments. In the third assignment, where students were assigned to create a game in Game Maker, some students did show some frustration. Because we assigned groups a random design document created by another group, some groups had to implement a type of game they had little or no interest in. After realizing a similar situation could and probably would happen if they entered industry, some students' notions about working in the game industry were tarnished. Other students appeared shocked at the amount of work that had to be put into such a simple-looking game. There were also students who seemed to gain respect for the industry, and how well everything had to fit together to make a presentable product.

Students also expressed some frustration during the text-based adventure assignment. After creating a playable game in Game Maker, it seemed that most students assumed creating a text-based adventure, in a program that used English sentences instead of more traditional syntax, would be easy. Students found that many words in the English language have multiple interpretations, and often commands functioned differently than they had expected. To successfully complete their text-based adventures, most students had to spend hours reading through many pages of the Inform documentation. Many of the students with previous programming experience seemed to find the shift away from more traditional programming syntax very retrogressive.

6.2 Fall 2009 offering

Although students performed well on both game projects, the positive change in student effort towards their work during the second project was apparent. It may have been due to the fact that the first game project was completed without any peer review activities, whereas there was an abundance of peer review activities throughout the second. Another thing to consider is that the first assignment was not worth as many marks as the final project, as the first assignment was intended to be a learning experience for the students. Although all students received a passing grade on the first assignment, there was a noticeable difference in game quality between groups during the first assignment game presentations. Students quickly realized it was not fun to present a game they were not proud of to the whole class. This can serve as a motivating factor - for instance, the programmer for a group that received a lower mark on the first project had a working prototype for his final project the day after the project was assigned.

Students who did well on a game presentation seemed to achieve a sort of special social status. Members of groups that had given a strong presentation earned a lot of respect from their peers, and their opinions were highly regarded and sought after in subsequent classes and lab sections. Further evidence towards some sort of social stratification was apparent when the time came to form new groups for the second project, as members of these “successful” groups were specifically requested.

All students worked very hard to impress their peers in their game prototypes and final presentations. Being recognized as someone who provided useful feedback also seemed important to students. I created a forum to list the most useful peer comments, and this seemed to further motivate students to provide helpful feedback to other groups. Students seemed very proud when someone posted their piece of feedback as the best feedback they had received during that session; they seemed pleased they could so significantly contribute to another project in just a single peer review session. It would be interesting to further study the indirect effects peer review activities have on students, and how it can be used to motivate students and strengthen the sense of community within a course. In our course, these peer review activities speak towards a burgeoning sense of community, competition, and excitement that seemed to emerge because of the peer review activities, as evidenced by the responses to three of the survey statements.

In the final weeks of the course there was an abundance of peer review activities, which could have had some effect on how students responded to the statement on the survey regarding more peer review in the course. The results for this statement do not necessarily mean students did not enjoy or feel the peer review activities were effective, just that they did not want a higher number of peer review activities. Unfortunately, the survey did not query the students in regards to whether they felt that the class should have had less peer review activities. The survey statements that received the highest overall average suggests that students found that the peer review activities were helpful: peer review activities got students thinking more about their own games, and studends felt the activities helped bring up ideas they did not think they would have considered themselves. For example, students began using the peer review activities as a sort of learning tool: often during presentations students discussed how and where they were stuck, and used the peer reviewers as a source of information. Furthermore, students discussed multiple game design possibilities during their presentations, and used the peer feedback to help make certain difficult design decisions. Students also felt that their first game projects could have been improved had we incorporated peer review activities into the project. As mentioned in the previous section, many students chose to attend multiple lab sessions during the last few weeks of the course (all members of a given group were enrolled in the same lab section). This allowed a group to give multiple prototype demonstrations to students they would not normally receive feedback from, and they also participated as peer reviewers for those students, increasing the amount of feedback for everybody. In summary, it seems that although the majority of students may not have wanted more peer review activities, they nonetheless recognized that the peer review activities were a valuable asset to their projects.

Some students were comfortable participating in class discussions right from the beginning of the semester. I hoped that adding peer review activities into our course would encourage all students to more actively participate in class discussion. Overall participation in class discussion did seem to increase throughout the semester. However, I noticed that students who were less inclined to speak in class discussions at the start of the semester, still participated less than other more outgoing students during the final few weeks of the semester. So while I found that peer review seemed to foster a greater sense of community, this did not lead into more participation in general class discussion for many of our students.

Other researchers found that students felt uncomfortable or unqualified when doing peer review activities. We provided practice activities to ready students for the peer review process, but our survey results suggest that students did not feel that all of our preparation activities were necessary. In particular, the students felt capable to give feedback for the first game project presentations in lieu of the instructors. However, it should be noted that students were evaluating their feelings in retrospect, and it may not have been representative of exactly how they felt initially. Some of the art and writing students had prior peer review experience from other courses offered outside of computer science. The students' level of comfort could also have been due to the subjective nature of our peer review activities. Other courses employ peer review activities on *code* [53, 62, 35], in which providing feedback may be intimidating for first year students, especially those new to programming. We did not perform peer review activities on game code and implementation specifically, but on the design and playability. I did notice that during the later peer review sessions, implementation methods were very often discussed, by this time students may have been a little more comfortable discussing implementation issues. The above reasons form the basis of why I think that a studio-based learning approach can so easily be integrated into a course on game design, it is because many students are quite comfortable with games in general. Students in our course preferred games from a wide variety of genres, and students were encouraged to give honest personal opinions based on their own game experiences. Although I provided rubrics for the peer review, I stressed that there is no single perfect game design idea.

Another problem researchers had identified with peer review was conspiracy activities. I did not find any indication of conspiracy throughout our peer review tasks, possibly due to the fact that students could not directly affect their peers' grades. The emphasis of our peer review activities was on providing constructive feedback rather than pointing out flaws, negating any reason for students to provide dishonest feedback. The forum post containing the most helpful peer comments may have also helped motivate students to give effective feedback.

According to our surveys, students found the peer review activities got them thinking more about their own games and made them more confident going into their final presentation. The discussions about implementation methods during the last few weeks of the semester further suggest that the peer review activities promoted self reflection. These results are a positive but not unexpected outcome of our peer review approach. In the second project, students were forced to get started on a prototype right away, as they were required to make prototype presentations. The feedback received in the peer review activities allowed groups to refine many different aspects of their games before the final presentation. Reviewing other groups' game prototype presentations promoted self reflection, and a critical mindset that they could apply to their own work. Giving prototype presentations was good practice, and the feedback from peers helped students gauge their readiness for the final presentation.

Chapter 7

Conclusions

In order to address industry issues in our multi-disciplinary course on game design, we started by combining several features from other games courses that had yielded successful results. We then augmented these features by constructing activities that pivoted largely on assessments valued by industry: cooperation and communication within teams, peer evaluation, and navigation and problem solving within pre-existing infrastructures. Our games course was first offered at the University of Victoria in the Fall 2008 semester, and a direct comparison to a more traditional course offering revealed that not only was the game design theme attractive to non-CS majors, but it also inspired them to consider taking more computer science courses. Unfortunately, our cooperative learning approach appeared to have a negative effect on some of our students' interest in pursuing further studies game design. Many students did not see the value in some of our collaborative activities, and were not comfortable participating in them. We outline the significant changes we made to our course activities for the Fall 2009 offering of the course in order to improve the learning experience for our students. The main refinement to our previous course was the integration of a studio-based learning approach into our final game project: students were provided with the opportunity to give and receive feedback at multiple stages throughout the development of their games, instead of only during final presentations. Studio-based learning greatly enhances the perceived effectiveness of peer review, as students are able to apply changes to their work based on the feedback they received from their peers before being graded. The educational benefit to students is that this allows them to explore multiple implementation methods to a given problem, and receive insight and suggestions about possibilities they may not have considered themselves. As peer reviewers, students are exposed to a variety of different game solutions, and the process facilitates self reflection. During the Fall 2009 offering, we observed some interesting results regarding how these activities affected student social status, sense of community, competition, and motivation to perform at the top of the class. An exit survey suggests that our refined peer review activities may have had a positive effect on student engagement, and that students appeared to feel that the peer review activities were effective in helping them improve their game projects.

Previous researchers have found that some students are apprehensive about doing presentations and peer review activities. After allocating time in the lectures and labs to prepare students for these types of activities, it appeared that students were much more comfortable with the peer review process. Traditional computer science assignments often have just one particular right solution, and so collaboration on assignments is not often allowed. There were a wide variety of games created in our course, and therefore students were extremely limited in the amount of material they could copy from another group. In game design there is also no single or right solution, so conspiracy activities were not an issue in our course, and we actually encouraged discussion on different possible solutions when problems arose during project work. Students were only graded on participation, and were encouraged to give honest feedback based on their personal game preferences and backgrounds. The subjective nature of game design allows students to give feedback without fear of being incorrect, and it is this subjective nature that makes a studio-based learning approach fit so well into a course on game design.

Bibliography

[1] *Job Outlook*. National Association of Colleges and Employers (NACE), 2006.

[2] Mohammed Al-Bow, Debra Austin, Jeffrey Edgington, Rafael Fajardo, Joshua Fishburn, Carlos Lara, Scott Leutenegger, and Susan Meyer. Using game creation for teaching computer programming to high school students and teachers. In *ITiCSE '09: Proceedings of the 14th annual ACM SIGCSE conference on Innovation and technology in computer science education*, pages 104–108, New York, NY, USA, 2009. ACM.

[3] Lecia J. Barker. When do group projects widen the student experience gap? In *ITiCSE '05: Proceedings of the 10th annual SIGCSE conference on Innovation and technology in computer science education*, pages 276–280, New York, NY, USA, 2005. ACM.

[4] Lecia Jane Barker, Kathy Garvin-Doxas, and Michele Jackson. Defensive climate in the computer science classroom. *SIGCSE Bull.*, 34(1):43–47, 2002.

[5] Tiffany Barnes, Heather Richter, Eve Powell, Amanda Chaffin, and Alex Godwin. Game2learn: building cs1 learning games for retention. In *ITiCSE '07: Proceedings of the 12th annual SIGCSE conference on Innovation and technology in computer science education*, pages 121–125, New York, NY, USA, 2007. ACM.

[6] Christine Bauer, Kathrin Figl, Michael Derntl, Peter Paul Beran, and Sonja Kabicher. The student view on online peer reviews. In *ITiCSE '09: Proceedings of the 14th annual ACM SIGCSE conference on Innovation and technology in computer science education*, pages 26–30, New York, NY, USA, 2009. ACM.

[7] 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.

[8] Leland L. Beck and Alexander W. Chizhik. An experimental study of cooperative learning in cs1. In *SIGCSE '08: Proceedings of the 39th SIGCSE technical symposium on Computer science education*, pages 205–209, New York, NY, USA, 2008. ACM.

[9] Leland L. Beck, Alexander W. Chizhik, and Amy C. McElroy. Cooperative learning techniques in cs1: design and experimental evaluation. In *SIGCSE '05: Proceedings of the 36th SIGCSE technical symposium on Computer science education*, pages 470–474, New York, NY, USA, 2005. ACM.

[10] Andrew Begel and Beth Simon. Novice software developers, all over again. In *ICER '08: Proceeding of the Fourth International Workshop on Computing Education Research*, pages 3–14, New York, NY, USA, 2008. ACM.

[11] Andrew Begel and Beth Simon. Struggles of new college graduates in their first software development job. In *SIGCSE '08: Proceedings of the 39th SIGCSE technical symposium on Computer science education*, pages 226–230, New York, NY, USA, 2008. ACM.

[12] Benjamin S. Bloom. *Taxonomy of Educational Objectives: Handbook 1: Cognitive Domain*. Longman, New York, 1956.

[13] Lil Blume, Ron Baecker, Christopher Collins, and Aran Donohue. A "communication skills for computer scientists" course. In *ITiCSE '09: Proceedings of the 14th annual ACM SIGCSE conference on Innovation and technology in computer science education*, pages 65–69, New York, NY, USA, 2009. ACM.

[14] David Boud. Sustainable assessment: rethinking assessment for the learning society. *Studies in Higher Education*, 22(2):151–167, November 2000.

[15] Ruth Butler. Enhancing and undermining intrinsic motivation: the effects of task-involving and ego-involving evaluation on interest and involvement. *British Journal of Educational Psychology*, 58(1):1–14, February 1988.

[16] Donald Chinn. Peer assessment in the algorithms course. In *ITiCSE '05: Proceedings of the 10th annual SIGCSE conference on Innovation and technology in computer science education*, pages 69–73, New York, NY, USA, 2005. ACM.

[17] Kajal Claypool and Mark Claypool. Teaching software engineering through game design. In *ITiCSE '05: Proceedings of the 10th annual SIGCSE conference on Innovation and technology in computer science education*, pages 123–127, New York, NY, USA, 2005. ACM.

[18] Daniel C. Cliburn. The effectiveness of games as assignments in an introductory programming course. *Frontiers in Education Conference, 36th Annual*, pages 6–10, Oct. 2006.

[19] Daniel C. Cliburn and Susan Miller. Games, stories, or something more traditional: the types of assignments college students prefer. In *SIGCSE '08: Proceedings of the 39th SIGCSE technical symposium on Computer science education*, pages 138–142, New York, NY, USA, 2008. ACM.

[20] Douglas D. Dankel, II and Jonathan Ohlrich. Students teaching students: incorporating presentations into a course. In *SIGCSE '07: Proceedings of the 38th SIGCSE technical symposium on Computer science education*, pages 96–99, New York, NY, USA, 2007. ACM.

[21] Katherine Deibel. Team formation methods for increasing interaction during in-class group work. In *ITiCSE '05: Proceedings of the 10th annual SIGCSE conference on Innovation and technology in computer science education*, pages 291–295, New York, NY, USA, 2005. ACM.

[22] Tamara Denning, Michael Kelly, David Lindquist, Roshni Malani, William G. Griswold, and Beth Simon. Lightweight preliminary peer review: does in-class peer review make sense? In *SIGCSE '07: Proceedings of the 38th SIGCSE technical symposium on Computer science education*, pages 266–270, New York, NY, USA, 2007. ACM.

[23] Michael Eagle and Tiffany Barnes. Wu's castle: teaching arrays and loops in a game. *SIGCSE Bull.*, 40(3):245–249, 2008.

[24] Michael Eagle and Tiffany Barnes. Experimental evaluation of an educational game for improved learning in introductory computing. *SIGCSE Bull.*, 41(1):321–325, 2009.

[25] Christopher Egert, Stephen Jacobs, and Andrew Phelps. Bridging the gap: balancing faculty expectations and student realities in computer gaming courses. In *Future Play '07: Proceedings of the 2007 Conference on Future Play*, pages 201–204, New York, NY, USA, 2007. ACM.

[26] Anthony Estey, Amy Gooch, and Bruce Gooch. Addressing industry issues in a multi-disciplinary course on game design. In *FDG '09: Proceedings of the 4th International Conference on Foundations of Digital Games*, pages 71–78, New York, NY, USA, 2009. ACM.

[27] Anthony Estey, Jeremy Long, Bruce Gooch, and Amy Gooch. Investigating studio-based learning in a course on game design. In *FDG '10: Proceedings of the 5th International Conference on Foundations of Digital Games*, pages 64–71, New York, NY, USA, 2010. ACM.

[28] Career Explorer. Ten hottest careers: 2010. *http://www.careerexplorer.net/ten-hottest-careers.asp*, Retrieved May 20, 2010.

[29] D. Sluijsmans F. Dochy, M. Segers. The use of self-, peer and co-assessment in higher education: A review. *Studies in Higher Education*, 24(3):331–350, 1999.

[30] Rosemary Garris, Robert Ahlers, and James E. Driskell. Games, motivation, and learning: A research and practice model. *Simulation Gaming*, 33(4):441–467, December 2002.

[31] Graciela Gonzalez. A systematic approach to active and cooperative learning in cs1 and its effects on cs2. In *SIGCSE '06: Proceedings of the 37th SIGCSE technical symposium on Computer science education*, pages 133–137, New York, NY, USA, 2006. ACM.

[32] John Hamer, Helen C. Purchase, Paul Denny, and Andrew Luxton-Reilly. Quality of peer assessment in cs1. In *ICER '09: Proceedings of the Fifth International Workshop on Computing Education Research Workshop*, pages 27–36, New York, NY, USA, 2009. ACM.

[33] Jessen T. Havill and Lewis D. Ludwig. Technically speaking: fostering the communication skills of computer science and mathematics students. In *SIGCSE '07: Proceedings of the 38th SIGCSE technical symposium on Computer science education*, pages 185–189, New York, NY, USA, 2007. ACM.

[34] James M. Hogan and Richard Thomas. Developing the software engineering team. In *ACE '05: Proceedings of the 7th Australasian Conference on Computing Education*, pages 203–210, Darlinghurst, Australia, Australia, 2005. Australian Computer Society, Inc.

[35] Christopher Hundhausen, Anukrati Agrawal, Dana Fairbrother, and Michael Trevisan. Integrating pedagogical code reviews into a cs 1 course: an empirical study. *SIGCSE Bull.*, 41(1):291–295, 2009.

[36] Christopher D. Hundhausen, N Hari Narayanan, and Martha E. Crosby. Exploring studio-based instructional models for computing education. *SIGCSE Bull.*, 40(1):392–396, 2008.

[37] Roger T. Johnson and David W. Johnson. An overview of cooperative learning. In A. Nevin J. Thousand, A. Villa, editor, *Creativity and Collaborative Learning.* Brookes Press, Baltimore, 1994.

[38] Lisa C. Kaczmarczyk, Matthew R. Boutell, and Mary Z. Last. Challenging the advanced first-year student's learning process through student presentations. In *ICER '07: Proceedings of the third International workshop on Computing education research*, pages 17–26, New York, NY, USA, 2007. ACM.

[39] Robert M. Keller. *Computer Science: Abstraction to Implementation.* Harvey Mudd College, Claremont, CA, United States, 2001.

[40] Patricia Lasserre. Adaptation of team-based learning on a first term programming class. In *ITiCSE '09: Proceedings of the 14th annual ACM SIGCSE Conference on Innovation and Technology in Computer Science Education*, pages 186–190, New York, NY, USA, 2009. ACM.

[41] Learning and Teaching Centre, University of Victoria. Course (Re)Design Workshop. http://www.ltc.uvic.ca/events/courses/CRW.php, August 2008.

[42] 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.

[43] Jeffrey J. McConnell. Active and cooperative learning: further tips and tricks (part 3). *SIGCSE Bull.*, 38(2):24–28, 2006.

[44] Colleen McCreary. Living the dream. `http://www.academicresourcecenter.net/curriculum/pfv.aspx?ID=6947`, February 23, 2007.

[45] Lakshman Myneni, Margaret Ross, Dean Hendrix, and N. Hari Narayanan. Studio-based learning in cs2: an experience report. In *ACM-SE 46: Proceedings of the 46th Annual Southeast Regional Conference on XX*, pages 253–255, New York, NY, USA, 2008. ACM.

[46] David J. Nicol and Debra Macfarlane-Dick. Formative assessment and self-regulated learning: a model and seven principles of good feedback practice. *Studies in Higher Education*, 31(2):199–218, April 2006.

[47] 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.

[48] 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.

[49] David A. Patterson. Computer science education in the 21st century. *Commun. ACM*, 49(3):27–30, 2006.

[50] William G. Perry. *Forms of Ethical and Intellectual Development in the College Years*. Jossey-Bass, San Francisco, CA, 1999.

[51] Sarah Monisha Pulimood and Ursula Wolz. Problem solving in community: a necessary shift in cs pedagogy. In *SIGCSE '08: Proceedings of the 39th SIGCSE technical symposium on Computer science education*, pages 210–214, New York, NY, USA, 2008. ACM.

[52] P Ramsden. *Learning to teach in higher education*. Routledge, London, 1992.

[53] Ken Reily, Pam Ludford Finnerty, and Loren Terveen. Two peers are better than one: aggregating peer reviews for computing assignments is surprisingly accurate. In *GROUP '09: Proceedings of the ACM 2009 International Conference on Supporting Group Work*, pages 115–124, New York, NY, USA, 2009. ACM.

[54] Susan L. Reiser and Rebecca F. Bruce. Fabrication: a tangible link between computer science and creativity. In *SIGCSE '09: Proceedings of the 40th ACM technical symposium on Computer science education*, pages 382–386, New York, NY, USA, 2009. ACM.

[55] Debbie Richards. Designing project-based courses with a focus on group formation and assessment. *Trans. Comput. Educ.*, 9(1):1–40, 2009.

[56] Laurence Shatkin. *150 Best Recession-Proof Jobs*. Jist Publishing, 2008.

[57] Harald Sondergaard. Learning from and with peers: the different roles of student peer reviewing. In *ITiCSE '09: Proceedings of the 14th annual ACM SIGCSE conference on Innovation and technology in computer science education*, pages 31–35, New York, NY, USA, 2009. ACM.

[58] Nathan R. Sturtevant, H. James Hoover, Jonathan Schaeffer, Sean Gouglas, Michael H. Bowling, Finnegan Southey, Matthew Bouchard, and Ghassan Zabaneh. Multidisciplinary students and instructors: a second-year games course. In *SIGCSE '08: Proceedings of the 39th SIGCSE technical symposium on Computer science education*, pages 383–387, New York, NY, USA, 2008. ACM.

[59] Kelvin Sung, Michael Panitz, Scott Wallace, Ruth Anderson, and John Nordlinger. Game-themed programming assignments: the faculty perspective. *SIGCSE Bull.*, 40(1):300–304, 2008.

[60] Jay Vegso. Continued drop in cs bachelor's degree production and enrollments as the number of new majors stabilizes. *Computing Research News*, 19(2), March 2007.

[61] Henry M. Walker. Collaborative learning: a case study for cs1 at grinnell college and austin. *SIGCSE Bull.*, 29(1):209–213, 1997.

[62] Yanqing WANG, LI Yijun, Michael Collins, and Peijie LIU. Process improvement of peer code review and behavior analysis of its participants. In *SIGCSE '08: Proceedings of the 39th SIGCSE technical symposium on Computer science education*, pages 107–111, New York, NY, USA, 2008. ACM.

[63] Briana Lowe Wellman, Monica Anderson, and Susan V. Vrbsky. Preop as a tool to increase student retention in cs. *J. Comput. Small Coll.*, 25(2):167–175, 2009.

[64] Stuart Zweben. Computing degree and enrollment trends. *2007-2008 CRA Taulbee Survey*, 2008.

[65] Stuart Zweben. Upward trend in undergraduate cs enrollment; doctoral production continues at peak levels. *Computing Research News*, 21(3), May 2009.

[66] Michael Zyda, Victor Lacour, and Chris Swain. Operating a computer science game degree program. In *GDCSE '08: Proceedings of the 3rd International Conference on Game development in Computer Science Education*, pages 71–75, New York, NY, USA, 2008. ACM.