UNIVERSITY OF CALGARY

Exploratory Case Study: Experience With A Game Programming Assignment In An Introductory Computer Science Classroom

by

Jessica Celeste Mason

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES

IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE

DEGREE OF MASTER OF SCIENCE

DEPARTMENT OF COMPUTER SCIENCE, FACULTY OF SCIENCE

CALGARY, ALBERTA

SEPTEMBER, 2009

UNIVERSITY OF CALGARY

FACULTY OF GRADUATE STUDIES

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies for acceptance, a thesis entitled " Exploratory Case Study: Experience With A Game Programming Assignment In An Introductory Computer Science Classroom " submitted by Jessica Celeste Mason in partial fulfilment of the requirements of the degree of Master of Science.

Supervisor, Dr. Michele Jacobsen, Faculty of Education

Supervisor, Dr. Rob Kremer, Dept. of Computer Science

Dr. Christian Jacob, Dept. of Computer Science

Dr. Jennifer Lock, Faculty of Education

August 25, 2009

Date

Abstract

The relationship between a game programming assignment and the learning experience of students in first-year post-secondary Computer Science was evaluated through an exploratory case study. The purpose of this study is to gain an understanding of the meaning for those involved in a game programming assignment in a first-year computer science class. The findings from this case study can increase our understanding of the relationship between a game programming assignment and student interest, motivation, intention to remain in Computer Science, and self-efficacy.

The case study employed two questionnaires and an interview. Results of this study demonstrated a promising increase in student interest in the course, motivation, self-efficacy and intention to major in computer science after a game programming assignment. Exploratory results should be interpreted conservatively until studies confirming these exploratory results are completed. Promising findings encourage further study by Computer Science educators into teaching practice and the value of game programming assignments for learning.

Acknowledgements

I would like to acknowledge the tremendous assistance of my advisors Dr. Rob Kremer and Dr. Michele Jacobsen. Without the support offered to me by both Dr. Kremer and Dr. Jacobsen I would not have been able to finish my research and present this thesis. I would also like to acknowledge the contribution of my previous advisor Dr. Ken Loose. My mom and dad rock.

Table of Contents

List of Tables

List of Figures and Illustrations

Epigraph

"Sometimes we simply have to keep our eyes open and look carefully at individual cases – not in the hope of proving anything, but rather in the hope of learning something!"

(Hans Eysenck, 1976)

CHAPTER 1: INTRODUCTION

1.1 Aim

This exploratory case study provided an opportunity to better understand the relationship between game development assignments and student learning, motivation, interest and attitudes towards performance. This research provides an opportunity for empirical inquiry into game programming assignments as an effective tool for increasing post-secondary computer science students' motivation, interest, intention to continue in computer science and attitudes towards performance.

1.2 Background

The "Why"

Across North America, recent publications [Astin (2005), Vegso (2005), Vegso (2006), Vegso (2007)] report a new set of challenges in the discipline of Computer Science Education. Today's students belong to the gaming generation; they are described as "digital natives" [Prensky (2001)] with traits and experiences different from those of past generations or even the prior decade. The challenges faced by educators to encourage student interest, motivation, retention and self-efficacy are not new; however, they are contemporary challenges that cannot be ignored.

Games engage players in active participation (Papert, 1980). This thesis studies the relationship between a game programming assignment and student learning. Good games

can inherently promote engagement with a task and encourage repetition and mastery of new skills, not only for the game players but also for the game creators (Malone, 1980).

The use of games for learning is common practice in education and is evident throughout many learning arenas from very young children to post-secondary students. Games can be useful for learning because of their positive effect on interest and motivation (Macedonia, 2002). Game design and development experiences are similarly useful for learning. Within Computer Science, there are recent publications describing positive experiences using game design and development to interest and motivate students from introductory programming to capstone courses, for example Connolly (2005), Jones (2000) and Lawrence (2004).

Given this direction of emerging knowledge in the use of games and game programming assignments in post-secondary Computer Science, it is important to better understand the relationship between game programming assignments and student motivation, interest, retention and perceived performance, and whether this offers a promising direction for student learning.

1.3 Objectives

The "What"

The purpose of this research is to investigate whether game programming assignments are an effective tool for increasing post-secondary computer science students' interest,

motivation, intentions to remain in Computer Science and attitudes towards performance through an exploratory case study.

To better understand the nature of the relationship between a game programming assignment and a student's learning, this thesis defines and measures four constructs related to learning:

- Interest
- Motivation
- Intentions to remain in Computer Science
- Attitudes towards performance

Throughout this thesis, these components are known as the "4 Constructs".

This case study was designed to be an exploratory investigation. The study employed questionnaires and an interview to probe the four constructs. The study was guided by several research questions:

- if learners complete a game programming assignment, does this lead to greater interest in and motivation for their learning?
- if learners complete a game programming assignment, does this leads to a more positive perception of students' own performance?
- if learners complete a game programming assignment, does this affect students' intentions to major in Computer Science?

1.4 Research Overview

The "How"

Student interest, motivation, intention to remain in Computer Science and attitudes towards performance in an introductory computer science course are measured before and after a game programming assignment. In order to assess any changes to student learning, students were encouraged to submit two questionnaires of primarily Likert-scale questions immediately before and after the game programming assignment. An interview was conducted with the teaching assistant to supplement evidence from the questionnaires.

The responses were analysed to pursue the research questions of this thesis. The pre and post assignment questionnaires measured aspects of student interest, motivation, intention to remain in Computer Science and attitudes towards performance in the course and Computer Science as a discipline. In addition, the post-questionnaire also measured the amount of time students spent on their assignments, student impressions of the assignment, and time students spent playing computer games. Student responses from before and after the game programming assignment were compared to measure any differences that may be attributable to the influence of the game programming assignment.

1.5 Thesis Structure

The thesis is structured to provide a deeper understanding of:

- *why* this investigation was undertaken and why the study should be considered relevant and significant;
- *how* this case study was undertaken, including the operational definition, study design and procedure; and,
- *what* was determined from the study: the results including an analysis and discussion of the significance of the conclusions.

CHAPTER 2: LITERATURE REVIEW

2.1 Overview

Literature was reviewed in four areas considered relevant to the study. This chapter organizes the review in four key categories:

1. Computer science undergraduate student enrolment and retention;
2. Student interest and motivation;
3. Use of games in education: game play and game development; and,
4. Case study design and applications.

This literature review provides a background to the research and describes the motivations propelling the research project (categories 1, 2 and 3). In this key way, this is not a standard literature review for Computer Science, and instead reflects more of an educational research approach. Since the research is undertaken as a case study selected by the researcher, this background provides a justification for why this case was selected as worthy of study. Category 3 provides an overview of related research in the area of game development assignments and their impact on learners while category four provides an explanation of case study design and applications in exploratory research.

2.2: Student retention and enrolment

The Computing Research Association (2007) conducts and publishes an annual survey of Computer Science (CS) departments in North America known as the Taulbee Survey. Vegso (2006) presents the findings of the Taulbee Survey and claims that the number of

new CS majors has declined by 70% between Fall 2000 and 2005. The total number of bachelor's degrees granted in Computer Science declined 17% between 2003 and 2004. Vegso (2006) provides no analysis or discussion of the causes for this decline in enrolment and degrees granted in computer science nor does the paper offer any solutions or discuss the role of student interest and motivation in the retention of computer science majors. The study was repeated by Vegso (2007) a year later; he found that a further decline in undergraduate enrolment and degrees granted was observed in 2006.

Vegso (2005) provides an analysis of survey results from the Higher Education Research Institute at the University of California at Los Angeles. This survey indicates that the popularity of computer science as a major among incoming first-year students in the United States has dropped significantly in the past four years. The survey indicated that the percentage of incoming undergraduates planning to major in CS has declined by over 60 percent between the Fall of 2000 and 2004 and concludes that there will be a sharp decline in the number of bachelor's degrees granted in CS in the coming decade, not attributing any of the decline to a cyclical nature of the data. Astin (2005) considers that intentions to major are "a powerful predicator of retention."

A review was conducted to determine whether this cyclical pattern was observed in Canada. It was found that the number of undergraduate Computer Science (CS) graduates appears to follow a similar cycle of increase and decline at the University of Calgary (University of Calgary, 1998-2009) as shown in Figure 1. There are many possible explanations for this pattern. Research on trends across North America

(Lomerson and Pollacia, 2006) demonstrates that this cyclical pattern is not specific to the University of Calgary and presents a significant problem since demand for CS graduates has remained steady, and may have also increased, in recent years. "This may be a cyclical process in the information systems job market. As the economy comes back and the demand for IT workers resumes, businesses will find a significant shortfall in skilled workers" (Lomerson et. al., 2006, p. 4).

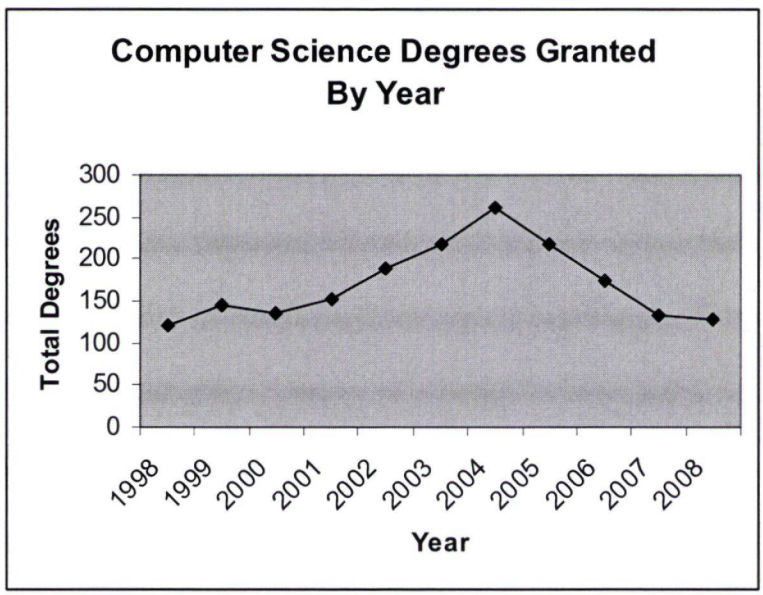

Figure 1: Undergraduate computer science degrees granted by year at the University of Calgary

The increases, decreases, or cyclical nature of enrolment in Computer Science may not be solely or significantly attributed to instruction. For example, Lomerson et. al., 2006, points to a strong relationship between the economy and degrees granted in Computer Science. Still, Beaubouef and Mason (2005) builds on Vegso (2006) and Astin's (2005) findings and investigates the possible causes for low retention rates of post-secondary students in Computer Science. Beaubouef et. al (2005) claims that North American

institutions are regularly reporting drop rates as high as 30 to 40% during the first and second year of courses, which may have something to do with instruction. Much of the discussion focuses on CS101 and CS102 and describes several possible causes of the poor retention of students. Beaubouef et. al (2005) speculate on causes including: poor advising or guidance of students while deciding their majors; poor math skills and problem solving abilities; poorly designed CS101 lab courses; lack of practice and feedback; low quality of teaching by graduate students; poor project management skills; and, teaching objects early or teaching objects late. Other than a very brief discussion on when to introduce objects, the authors do not address student interest, motivation, or self-efficacy as an investigable cause, nor even the content of the courses. The authors do not suggest any solutions but recommend further study into any factors affecting retention rates as an effort towards reducing the low retention rates. This case study is an effort to address the gap in our understanding about the relationship between student interest, motivation and self-efficacy and retention in computer science programs.

Becker and Parker (2005) offers suggestions for ways to encourage and retain undergraduate enrolment including non-major service courses and a focus on multimedia including gaming and entertainment. At the instruction level, the author recommends increased attention to student interests. Bergin and Reilly (2005) studied fifteen factors and their influence on the performance of students in a first-year programming course in 2003-2004. These factors included prior academic experience, prior computer experience, self-perception of programming performance, comfort level on the module

and specific cognitive skills. Bergin et. al. (2005)'s findings included a significant effect of student comfort and motivation on success in learning to program.

2.3 Student interest and motivation

According to Bandura (1986), one explanation for why people succeed is their perceptions of how skilled, competent and efficacious (having high self-efficacy) they are. Bandura (1977) believes that one's self-efficacy is determined by: one's perceived ability to succeed based on one's previous successes, the successes and failures of others, verbal persuasion (encouragement or discouragement from others), and emotional arousal. A student's emotional arousal is largely affected by his or her interest and motivation in the material. Bandura's much-cited work shows that a student's interest and motivation can have a large effect on his or her successful learning.

Alexander, Clark, Loose, Amillo, Daniels, Boyle, Laxer, and Shinners-Kennedy (2003) combined seven case studies of undergraduate requirements for admission to Computer Science programs across North America and focused particularly upon qualification entry: subjects studied in early university and student grades. According to Alexander et. al. (2003), not one of the admittance criteria studied, on which incoming students to postsecondary Computer Science are judged, are good predictors of student success in the study of computer science. For example, a student's previous grades did not have a significant correlation with the student's success in learning to program. This study was conducted across seven universities in North America and Europe, including the

University of Calgary. While Alexander et. al. do not list other possible factors related to student success at learning to program, the study reinforces the importance of studying factors which may affect student learning. Student interest and motivation, since not measured as admittance criteria, were not measured in the study. This result begs the question: what factors do contribute to student success at learning to program?

Bergin et. al. (2005) describes a study carried out in the 2004-2005 academic year examining the role of motivation and comfort-level in a first year development course. The study defines comfort-level as a student's ease with development, self-perception of their performance and self-efficacy for programming. The study found a strong correlation between intrinsic motivation and development performance. The study also found a strong correlation between self-efficacy for learning and actual programming performance. A regression model attributed 60% of the variance in programming performance to student motivation and comfort. This is further supported by Wilson (2002), who conducted a similar study to determine factors that promote successful learning in introductory computer science courses. The study revealed three predictive factors: comfort level, math background and attribution to luck (negative influence).

Bergin et. al. (2005) provides a review of other studies finding that students who are more intrinsically motivated than extrinsically motivated perform better and that students who have positive impressions of a subject tended to be both intrinsically and extrinsically motivated. Support included Roberts (2000) who added that students are more successful when there are opportunities to go above and beyond assignment

requirements. Furthermore, Bergin et. al. (2005) describes Mitchell's (2000) study that showed that students who feel they have a strong motivation for studying a subject have a more positive perception of the subject and about the amount of practical work involved.

Wilson and Shrock (2001) also discusses factors that influence development success and observes that the most important predictor of students' performance in introductory computer science course was comfort level, determined by the degree of anxiety a student felt about his or her performance and ability to succeed in the course. This is supported by Haden (2006) who notes that "the inherent difficulty in drilling students in programming basics is that it can be deadly boring" (p.81). Students who are not engaged in course material are students who probably will not learn the material (Lobo, Baliga, Bergmann, Stone and Shah, 2000).

The literature reviewed on interest and motivation suggest that there is a gap in our understanding of how these two constructs relate to student success in a programming assignment. This case study is designed to offer some new understanding of this relationship between interest, motivation and a game programming task.

2.4 Use of games in education: game play and game development

> "Games are a common form of playing. All games have properties, rules and procedures that must be mastered in order to become a 'player'."
> (Rosas, Nussbaum, Cumsille, Marianov, Correa, Flores, Grau, Lagos, López, López, Rodriguez, and Salinas, 2003, p. 72)

The practice of using games for learning is being seen in mainstream culture as an acceptable educational strategy. The modern body of research into games for learning, in both traditional and digital formats, was led by *Mindstorms* (Papert, 1980). Seymour Papert encouraged research and practice within games for learning by indicating that students perform higher learning when they are more engaged with the learning. Students engage with games: they are interested and motivated by their play. Papert claims that both game play and game programming assignments engage students.

While focusing on the impact of new technologies on learning, Papert developed "Logo", a programming language for creating and modeling simulations while playing. He illustrates the difference between game play and game development in his book *The Connected Family* (1996). He explains that an instructionist approach to the classroom might employ a game to teach students a specific skill. In contrast, a constructionist approach to using game programming presents students with the challenge of inventing and creating the game themselves.

> "The scandal of education is that every time you teach something, you deprive a child of the pleasure and benefit of discovery (Papert 1980, p. 68)."

According to Papert, learners build their own cognitive tools or understanding of concepts and solutions through actively working with problems and by building external artefacts. Knowledge is not considered a commodity that can be transmitted and

retained; knowledge is gained by the learner constructing and reconstructing knowledge through personal experience. This is the theory of constructivism, formalized by Jean Piaget (1950). Seymour Papert (1980) additionally states that constructionist learning happens especially well when people are engaging in constructing a finite product, "such as a sand castle, a machine, a computer program, a book (p. 3)." This is the theory of constructionism – learning through building objects to think with. Kafai (1995), a graduate student of Papert's who built upon his constructionist theory, emphasised that valuable learning experiences are found when the activity is personally meaningful to the learner and builds upon the learner's own prior knowledge and interests. Programming games make learning more meaningful to students (Papert, 1980). Game development assignments fit the criteria for a constructionist learning experience with the game being the finite product built by the learner. Game development assignments can be particularly meaningful as a constructionist learning experience since learners are familiar with games, and are building on their past experiences in order to create something new.

Prensky (2001) has taken the position that today's students "think and process information fundamentally differently from their predecessors (p. 1)." Prensky refers to these students as "Digital Natives": native speakers of the digital language of computers, video games and the Internet. The speed of learning and time focusing on a single topic are distinctive characteristics of Digital Natives, as described by Prensky (2001).

In this next section, the prevalent use of games for learning across contexts will be explored to gain an understanding of the benefits of these approaches to learning and the possible relationship with learning to program games. Many instructional games, such as typing games, drill students on specific knowledge outcomes. Knowledge outcomes are assumed within more advanced and successful games as rules or conditions within the games that must be met for the player to be successful. Egenfeld-Nielsen (2004) describes examples including Packy & Marlone (1997) and Bronkie the Bronchiasaurus (1994). In the game, Bronkie the Bronchiasaurus (1994), the player controls a dragon to fight evil dinosaurs, build a clean air machine and properly manage asthma. Packy & Marlone (1997) addresses diabetes self-care among children by requiring the player to monitor blood glucose levels, take insulin injections and choose appropriate foods. Empirical studies indicate that both games effectively improved players' self-efficacy and specific knowledge. As a result of Packy & Marlone (1997), the experimental group showed a 77% drop in visits to urgent care.

The Armed Forces of the United States of America is motivated to capture the interest and attention of young people. The research of the American Army finds that the attention span of youth is much shorter than that of previous generations and that there is a distinct shift in focus of learning from passive transmission to discovery-based experiential and example-based learning. In 1999, the American Army established the Institute for Creative Technology (ICT), focusing on the development of simulator games with the purpose of training combat soldiers, motivating and interesting youth for recruitment to the Armed Forces, and retaining soldiers by engaging them in their roles.

(Macedonia, 2001). Games are also built and used by the Armed Forces for advanced skill-based training and practice. Troops use games, or simulations, to "practice exhaustively, taught by simulators not only how to use their ever more complex equipment, but also how to work in teams, move efficiently through a battlespace, and negotiate a wide range of conflicts." (Macedonia, 2002) The games are credited with reducing casualties in Iraq, the Balkans and Afghanistan according to a task force of the U.S. Defense Science Board.

The use of games for learning includes the use of games for persuasion. Games have been used to advertise wrinkle cream (Reversa, 2007) because of their effect on buyer interest and motivation. According to Vedrashko (2006), advertising placed within games is more persuasive than advertising placed within television and other medias. "Players demonstrate stronger recall and purchase intent after being exposed to brands during game sessions" (p. 23).

Just as video games can engage and train soldiers and consumers, video games can interest and intrinsically motivate learners (Malone, 1980). Learners are intrinsically motivated when finding an activity rewarding for its own sake rather than for the sake of an external reward. Malone (1980) describes the essential characteristics of video games that foster quality learning. These characteristics focus around goals and feedback, thereby pursuing intrinsic motivation by building player interest and self-efficacy.

At the undergraduate computer science level, Lawrence (2004) notes that "games capture student interest because they are fun and exciting, and students tend to learn more when actively engaged by the subject" (p.1). Students are familiar with games.

> "Current wisdom implies that learning is more effective when we build on what the learner already knows, and using situations they are familiar with." (Becker and Parker, 2005, p.2)

The positive effects of computer games used as instructional tools include strengthened academic achievement, student self-efficacy, and motivation towards learning (Rosas et. al, 2003). Very important for successful computer scientists, playing video games also favours the development of complex thinking skills related to problem solving and strategic planning (Rosas et. al, 2003). Designing learning experiences around these prior gaming experiences has the potential to take advantage of the students' particular problem solving and strategic planning abilities developed from their experience playing video games.

Research on game programming experiences suggests that leaving game development to the learners leads to better learning (Kafai, 1995). "In some senses, computer programming itself is one of the best computer games of all. In the 'computer programming game', there are obvious goals and it is easy to generate more. The 'player' gets frequent performance feedback (that is, in fact, often tantalizingly misleading about the nearness of the goal)" (Malone, 1980, p. 168).

Specifically within undergraduate computer science education, game development assignments are used to interest and motivate students from introductory courses to game development capstone classes (Parberry, Roden, and Kazemzadeh, 2005). DeLaet, Slattery, Kuffner, and Sweedyk (2005) describe several ways of teaching game development in their undergraduate programs across four universities in the United States. Lawrence (2004) describes an experience using a game development assignment in a second year data structures course and its effect on students' perceptions of their learning and interest in the assignment. The game development assignment was considered by the students to be the most interesting computer science assignment and increased student interest in the course, according to a class survey. Additionally, the majority of learners reported in the survey that they felt the assignment helped them become better programmers, and that it positively affected the self-efficacy of the learners.

The findings of Lawrence (2004) are supported by many informal studies following the use of game development assignments in introductory classes. Valentine (2005) describes an experience using Reversi in CS102 and its effect on student interest and motivation. Valentine was gratified to see students excited to create a high-quality playable game, and reported improved self-efficacy among learners as they "bragged" to peers and other faculty about the quality of their creation.

Ladd (2006) describes how "programming a text adventure game challenges and motivates students in a project-based CS1.5 course. It also provides opportunities to

introduce topics as varied as the history of computers, the fundamental separation of data and computation, and writing clear, concise prose (p. 163)." Ladd (2006) "leverages that widespread interest in computer games while keeping the focus of the course clearly on the fundamental concepts of computer science (p. 163)." Ladd (2006) reported that student engagement within the discipline improved as a direct result of the game development assignment: students were more motivated to complete the work necessary to learn.

Connolly (2004) was motivated to use a game development project to help motivate students to learn given the large amount of literature supporting the use of games for education and by providing an assignment that was more relevant to the lives of his students than a business application. Connolly (2004) was also encouraged by the experience of Jones (2000) who noted that games can provide "an extremely project-oriented, upper-division course to exercise and enhance the development and problem-solving skills of advanced students (p. 260)." Feedback from Connolly's students was mostly positive; however, student complaints included that the game assignment required them "to think constantly" and that it required them to use all their knowledge from previous courses.

The experience of Connolly (2004) seems to further support Jones (2000) who claimed that the "integration of concepts and techniques required to design and build computer games covers many of the topics offered in an undergraduate computer science

curriculum, allowing students concrete application of much of the theory, concepts, and skills they have been exposed to (p. 260)."

Leska and Rabung (2005) also describe an experience using game development assignments in introductory computer science courses to capitalize on student desire for graphical output and feedback. The graphical output of the game assignment both increased the quality of immediate feedback for the learner and also provided a concrete and quantifiable goal for the learner. The availability of rich feedback and goals are characteristics of video games that can interest and intrinsically motivate learners (Malone, 1980). The game development assignment for Leska et. al. (2005) also found that this fostered quality learning within an introductory computer science course by building player interest and self-efficacy.

Based on classroom experience with computer science undergraduate students who programmed games, Becker (2001) argues that with games, "students have a clear and identifiable goal and because they already know how it's supposed to work and find the task at hand interesting, they are more motivated to actually get their programs running, not just for the marks but also for their own satisfaction (p. 26)." Becker (2001) also notes that games appear to provide the opportunity to keep advanced students interested and motivated and challenged on more difficult "bonus" components of the games. This case study is an attempt to study the relationship between interest, motivation and a game programming assignment using empirical means.

Roberts (2000) addresses the major challenge of teaching introductory computer science by providing interesting and challenging assignments that are open-ended and have a development competition aspect. These assignments motivated students and kept them interested so that Roberts (2000) reports "seeing highly talented students get so turned on by computer science and the challenge of working on open-ended problems that they learn more in a week than many of my students do in the entire term (p. 299)." Designing assignments to motivate and interest students can have a profound impact on the level of learning undergone by the student.

In addition to engaging learners, Becker et. al. (2005) describes specific game development assignments and curriculum coverage for introductory courses. Combinations of games including checkers and arcade games were shown to cover the curriculum for CS2. This supports the experience described by Lawrence (2004) studying curriculum coverage of data structures in a first-year post-secondary computer science class using a game development assignment.

Multiple studies describe the effect of game programming assignment on student perceptions of their learning; however, there are no empirical studies evaluating the effect in a quantifiable method. Sindre (2003) reports a positive experience in providing students in introductory programming courses with a game programming assignment and states that "a topic for further work could be to give a proper scientific evaluation by means of performance test and/or questionnaires investigating student perceptions of their learning." Leutenegger and Edgington (2007), describing a similar experience that

learners found the assignment "fun" and "great", also herald the importance of a formal evaluation of the effect of game development assignments on student interest, motivation, retention and learning performance.

2.5 Case Study Design and Application

A case study provides the opportunity for empirical inquiry into a contemporary phenomenon. The phenomenon, the effect of game development assignments, is investigated within its own real-life context: the post-secondary classroom.

An exemplary case study, according to Yin (2003) must be significant, be complete, display sufficient evidence, consider alternative perspectives, and be composed in an engaging manner. For a case study to be significant, the phenomena being investigated must be worth studying; the researcher must make a reasonable argument for the importance of the case. Appropriate measures for the constructs being studied must be described and demonstrate validity for these instruments. For a case study to be complete, the completeness must be assured by using multiple sources of evidence, including existing literature on the phenomena. The evidence displayed must be sufficient. For a case study to consider alternative perspectives, the research must attempt pattern-matching, explanation-building and addressing rival explanations. The case study must also be engaging to its audience.

Not all case studies seek to establish a causal relationship or provide generalizable results. Cases studies may also provide an excellent opportunity for exploratory analysis. Exploratory case studies must still provide reliability of results by demonstrating that the methods employed in the case study can be repeated in a similar study. This reliability comes from a descriptive report of the case study application and method. When case studies are used in the preliminary stages of an investigation, this exploratory analysis may generate powerful hypotheses worthy of further investigation.

"Sometimes we simply have to keep our eyes open and look carefully at individual cases – not in the hope of proving anything, but rather in the hope of learning something!" (Eysenck, 1976, p. 9)

Within social sciences, performing research and learning in absence of "hard theory", means that proof can be hard to come by. Flyvberg (2006) defends the use of case studies against common objections that case studies cannot be generalized and contain a larger subjective bias. This misunderstanding about case study research is more common among natural science researchers. According to Flyvberg, these objections center on an inaccurate belief that case studies are less rigorous than experimental or quantitative methods. However, case studies do contribute to scientific development by providing a deep understanding of an individual case and are a disciplined approach to inquiry, a valid research method, commonly used in theory-building and theory-confirming research across disciplines.

Employing a case study approach to an exploratory research question allows greater opportunity to revisit and refine research questions throughout the exploration; hence, a case study offers a flexible and responsive research method for exploratory research. Conversely, at the start of an experimental study, phenomena are deconstructed into dependent and independent variables and almost the entire dialogue of ideas and evidence occurs through these secondary variables (Flyvberg, 2006). By generating and testing research questions, case study can be a valuable method for theory-building. "The advantage of the case study is that it can 'close in' on real-life situations and test views directly in relation to phenomena as they unfold in practice" (Flyvberg, 2006, p. 235).

With the purpose of exploratory case study research being to understand and learn more about the phenomena being studied, the investigation provides a close proximity to reality by letting the researchers place themselves within the context being studied. The case study provides an exploratory opportunity with the goal of better understanding the viewpoints and the behaviour making up the phenomena of a human-centred research question. The contextual application of a case study also provides the researcher with the opportunity to reflect upon observations. Before reporting the results of a case study, the researcher can refine and revise research questions. Reporting the findings of a case study brings to closure the results and findings of testing a research question within a specific context. In this way, the case study provides an excellent structure for exploratory research – for theory-building versus theory testing – and offering useful recommendations for the direction future research should take.

CHAPTER 3: METHODOLOGY

3.1 Research Objective

The objective of this thesis is to better understand the relationship between game development assignments and student learning, motivation, interest, attitudes towards performance and intentions to remain in introductory Computer Science.

During Stage 1 of the case study, student interest, motivation, intention to continue in Computer Science and attitudes towards performance (the four constructs) were examined *before* the game programming assignment to generate a baseline for comparison. Additionally, the results were analyzed to determine any relationships between the four constructs.

Stage 2 measured the four Constructs *after* the game programming assignment. Next, the results of Stage 1 were compared with the results of Stage 2 to examine any relationship between the game programming assignment and student interest, motivation, intention to continue in Computer Science and attitudes towards performance as measured by the questionnaire.

3.2 Case Study Design and Procedure

The "How"

In order to investigate the relationship between the game programming assignment and student interest, motivation, intention to continue in Computer Science and attitudes towards performance, initial student perceptions of their learning must be assessed during the Case Study. This exploratory case study design required the following: a population, a process and instruments for collecting data, and a justifiable method to analyze the data.

3.2.1 Participants

At a major University in Alberta, first year students are required to take CS101 (course codes and titles are replaced with the ACM curricula course code in order to mask the identity of the study's university) and CS102 in the ACM curricula. CS101, "Introduction to Computer Science I", is generally taken during the Fall Semester as a prerequisite to CS102. CS102, "Introduction to Computer Science II" is described in the University Calendar as

> "Continuation of Introduction to Computer Science I. The implementation of abstract data structures using objects, with emphasis on modularity and software design."

One section of CS102 was offered during the 2005-2006 regular school year in Winter 2006 with ninety-one students enrolled. Participants in this study were sampled from this population.

The course instructor of record has instructed undergraduate Computer Science for over twenty years, taught the studied course for more than 5 years, holds an M.Sc. in Computer Science, and actively researches education. Mid-semester there was a change in instructors to a Ph.D. student new to university instruction. The new instructor did not make any substantial changes to the course, followed the instructional plans of the previous instructor, and made no changes to the assignments or assessment of the previous instructor. The teaching assistant to the course, a graduate student, guided students in assignments during tutorials and marked all assignments.

The last assignment for the students was a game programming assignment. This provided an opportunity to investigate the effect of the game programming assignment on undergraduate introductory computer science students. The population sampled for this study is students in CS102 during the 2005-2006 regular school year at a major University in Alberta.

3.2.2 Sample

Although participation in the case study was voluntary, the importance of the study was emphasized to the students in order to obtain an accurate picture of the effect of the game programming assignment. Students were encouraged to submit both questionnaires by several announcements made in lectures, email announcements and reminders, as well as postings on Blackboard, the online learning management system. Both the pre and post questionnaires were administered entirely through Blackboard. There were no adverse consequences if subjects refused to participate in the survey. Students were informed that

their identity would be protected. All identifying information was removed immediately upon accessing the information from Blackboard and students were assigned non-identifying numbers to group responses from the pre and post questionnaires.

3.2.3 Procedure

Students were invited to complete questionnaires immediately before and immediately after the game programming assignment. These questionnaires were designed to measure the students' interest, motivation, attitudes towards performance and intentions to remain in Computer Science. In addition, the post-questionnaire also measured the amount of time students spent on their assignments, student impressions of the assignment, and time students spent playing computer games.

3.2.4 Game Programming Assignment

Throughout CS102, there were 5 assignments of increasing difficulty. Assignment 5 was a game programming assignment requiring the students to implement a game applying the curriculum concepts of class hierarchies and abstract classes. In past iterations of the course, this assignment was offered as either a game programming or as a business example, depending on the instructor. Students could work individually or in groups, in Java or C++. Students could choose between four different games: Asteroids, Frogger, Space Invader and Centipede.

1. Asteroids is a pioneer arcade game from the late 1970's of great popularity. The original version of Asteroids has alien spaceships and asteroids moving randomly around the screen. The player controls a spaceship (moving and shooting).

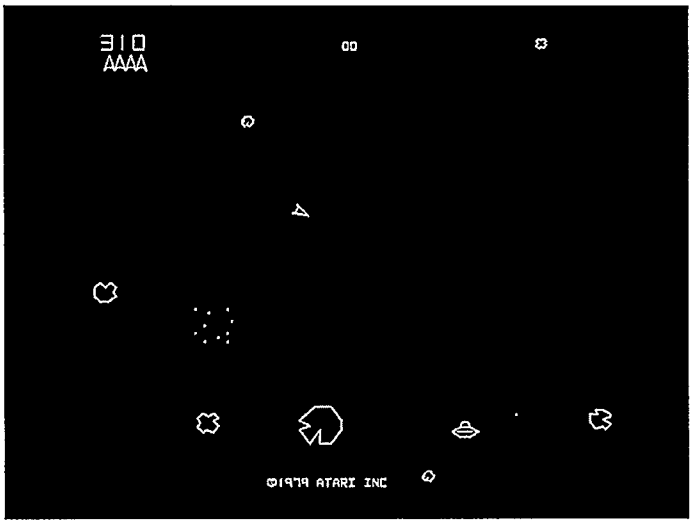

Figure 2: Asteroids Screenshot (Atari, 1979)

2. Frogger is another popular pioneer arcade game where the player controls a frog (moving and jumping). The player must dodge obstacles including traffic, other animals and a river to win.

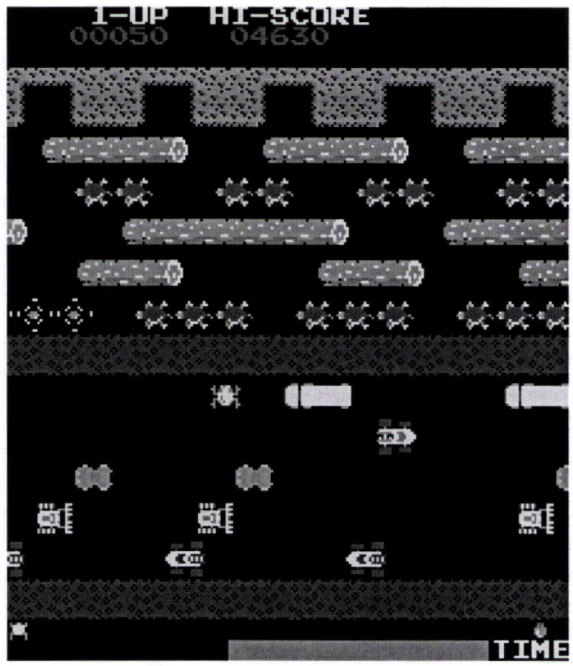

Figure 3: Frogger Screenshot (Konami, 1981)

3. Space Invaders remains one of the most popular arcade games ever made and was also created in the late 1970's. The player controls a shooter that can fly left or right along the bottom of the screen and shoot at rows of shooting aliens moving across the screen.

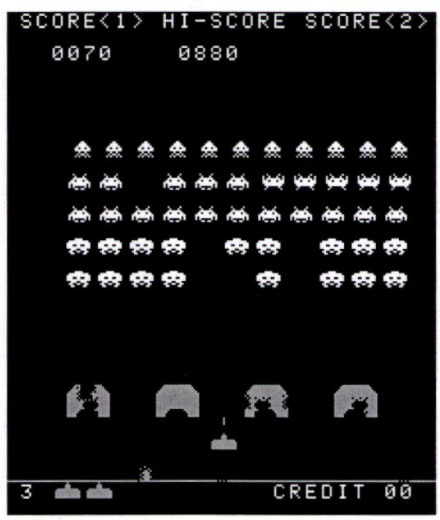

Figure 4: Space Invaders Screenshot (Taito, 1978)

4. Centipede was an arcade game created in 1981 that takes place in an enchanted mushroom patch where the player defends against insects by shooting a laser.

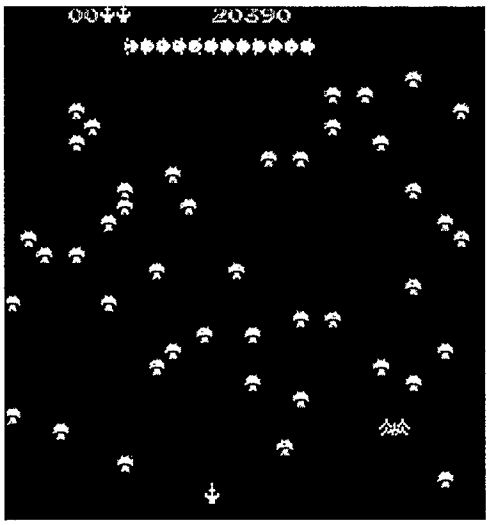

Figure 5: Centipede Screenshot (Atari, 1980)

There was no difference in terms of the curriculum coverage or difficulty level between the game choices. The course instructor, who specializes in the area of undergraduate computer science education, reviewed the assignments and determined that the assignments were roughly equivalent in terms of content coverage and difficulty. The reason for offering this choice was to give students options to program a game that most appealed to their interest or their previous game playing experience.

Assignment 5 was worth ten percent of the final course grade and was marked using a detailed rubric and a demonstration of their working game to the teaching assistant. This marking guide, Appendix 1, assigns marks on presentation, documentation, quality of

programming code, design, interface, and testing. Specifically to the curriculum goals of Assignment 5, students were marked on classes, subclasses, hierarchy, use of inheritance, overriding, overloading, polymorphism (abstract classes), exception handling, and event handling. The demonstration of a working game to the teaching assistant gave each student a chance to demonstrate specific features in their games. The teaching assistant used the rubric for marking during the demonstration. Bonus points were also given to students who went "above and beyond" and completed more difficult features in their games including graphical interfaces, sound effects and 3-D implementations. Using a rubric for assigning marks is an attempt to be more holistic by addressing all aspects of the creation of a game including design and development and demonstrating mastery of the curriculum goals.

3.2.5 Instruments

In order to assess the effect of the game programming assignment, the teaching assistant was interviewed and students were encouraged to submit 2 questionnaires:

1. the "pre" questionnaire completed immediately before Assignment 5
2. the "post" questionnaire completed immediately after Assignment 5.

The interview of the teaching assistant was an open-ended interview about the students' progress towards completion of the game programming assignment, their attitudes towards the assignment relative to the previous four assignments in the course, and their

performance on the marked assignments. Field notes were taken to capture key responses and ideas offered by the teaching assistant during the interview.

The primary constructs or areas of perceptions measured by both questionnaires were:

- attitudes towards performance and performance
- interest
- motivation
- intent to continue in Computer Science

For every construct, students responded to several questions relating directly to the construct. With the exception of student grades and expected grades, all questions regarding the four constructs were asked using the 5-point uni-dimensional Likert scale (strongly agree, agree, neither agree nor disagree, disagree, strongly disagree). Exact wording of each question is provided as part of section 3.3.

The instruments are unique since this is an exploratory case study without a pre-existing survey created from prior study. The questionnaire was not intended to ask questions confirming an understanding nor was the questionnaire designed as a testing instrument. The questions were framed to probe into the four constructs and were reviewed by an external assessor who was an experienced professor in computer science and researcher in computer science education including the use of experimental design and statistics. The external assessor agreed upon the face and content validity of the survey items.

In addition, secondary measures in the post-questionnaire were used for comparative analysis since the relationship is postulated to be important to the study:

- the amount of time students spent completing assignments
- student impressions of the game programming assignment
- student time spent playing video/computer games

Once again, responses were framed using the Likert scale except for responses of time measured in hours.

3.2.6 Data Analyses

The existing, or baseline, levels of student performance, interest, motivation and intention to continue in computer science were gathered from the pre-questionnaire. Responses were analyzed using descriptive statistics including frequency distribution and means. Strong, positive relationships within subscales were calculated using correlational analysis to identify the strength of grouping responses by construct (interest, motivation, intention to continue in computer science and attitudes towards performance) with significance established at $p<.05$.

As part of the second stage of data analysis, the initial data of the pre-questionnaire was compared to the data collected in the post-questionnaire. The variable of interest, or intervention point of the study, is whether or not the student completed the game programming assignment. As this variable is attributive and not controlled by the

investigator, formal experimental design was not an appropriate quantitative research approach. All students completed the game programming assignment; therefore, no control group was possible within the classroom.

Descriptive statistics and *t*-test analysis of variance were used to analyze and to compare responses before and after the assignment. Significant differences within a construct, for example between student interest after the game programming assignment and student interest before the game programming assignment, were determined using *t*-test analysis of means with the sample being every student from our population who completed both questionnaires. Students who completed only one questionnaire were not included in the comparison of means (t-test).

After the post-questionnaire was completed, the independent variable can be compared across two levels: before completing the game programming assignment and after completing the game programming assignment. Since the analysis is now between two groups, the quantitative research approach is comparative. This approach differs from the randomized experimental and quasi-experimental approaches because there is no active independent variable and the investigator cannot randomly assign participants to the groups. Hence, the second stage of research examines the presumed effect of the attribute independent variable by analyzing *t*-test results for significant differences. The significant difference is determined that the variance between groups, the treatment variance, is significantly larger than the variance within groups, the error variance.

This comparative design was used to determine any significant differences in student interest, motivation, intention to continue in Computer Science and attitudes towards performance. Significance level for all t-tests was established at $p<.05$.

Profile analysis was also performed during the second stage of data analysis to determine if any one construct could predict significant variance. These results were plotted by groups formed by the profile analysis.

Students were placed into sub-groups based on independent attribute variables:

- *Motivation:* high motivation vs. low motivation
- *Engagement:* length of time spent on Assignment 5, length of time spent on Assignment 5 relative to other assignments in the course, and enjoyment of Assignment 5 relative to other assignments in the course

The purpose of analyzing students in sub-groups was to provide a comparison and remark on significant differences attributed to secondary measures (dependent) that may explain the impact of the game programming assignment. T-tests were also used to determine if the game programming assignment was related to differences between our sub-groups with significance established at $p<.05$. Only students who completed both questionnaires could be included in this analysis.

The *Motivation* sub-group analysis compares the effect of the game programming assignment on highly motivated students against the effect of the game programming

assignment on less motivated students. The two groups are defined using the average response across all motivation questions from the pre-questionnaire:

- Highly motivated students: top 50% of respondents
- Less motivated students: bottom 50% of respondents

T-Test results may identify any significant differences in the effect of the game programming assignment based on student motivation.

The *Engagement* sub-group analysis compares the effect of the game programming assignment on highly-engaged students against the effect of the game programming assignment on less-engaged students. The two groups are defined using the average response across these three questions:

- How many hours did you spend completing Assignment 5?
- I spent more time completing Assignment 5 than previous assignments in CS 102 (CS102).
- I found Assignment 5 more enjoyable than previous assignments in CS 102 (CS102).

Highly engaged students: top 50% of respondents

Less engaged students: bottom 50% of respondents

T-Test results may identify any significant differences in the effect of the game programming assignment based on student engagement with the Assignment 5.

3.3 Operational Definitions

The "What"

Before measuring the effect of a game programming assignment on learners, interest, motivation, intention to continue in Computer Science and attitudes towards performance are operationally defined. This is done using the relevant questions from the questionnaires, included below. Most of these questions were phrased so that a student could strongly disagree, disagree, neutral, agree or strongly agree.

3.3.1 Interest

Measured separately are a student's interest in the discipline of Computer Science and the same student's interest in an introductory Computer Science course. The participant is asked to independently assess and self-report his or her interest as pertaining to our operational definition of 'interest'. 'Interest' is not defined by the research; the participant responds from his or her own point of view and the response without constraint imposed by the researcher's definition. The student is also asked about having a positive orientation or impression of the subject matter.

Operational Definition of 'Interest'

- Discipline of Computer Science
    i. "I would rate my interest in computer programming as high"
    ii. "My impression of computer programming and problem solving is positive"
- Introductory Computer Science course

i. "I would rate my interest in CS 102 as high"

ii. "My impression of CS 102 is positive"

3.3.2 Motivation

Motivation can be difficult to measure accurately. There are two general types of motivation: intrinsic and extrinsic motivation. Intrinsic motivation exists when a student is both willing (consenting, inclined and ready) and desiring (wanting or wishing, as for something that brings satisfaction or enjoyment) to succeed. Extrinsic motivation exists when a student feels a need to succeed (has an obligation or necessity arising from his or her individual situation) or, similarly, an external compulsion to succeed (feels obliged or coerced by someone else such as a parent). Traditional student motivators, such as grades, are generally examples of extrinsic motivation. In this study, intrinsic and extrinsic motivation were measured separately, as self-reported by each participant.

On a second dimension, a student's motivation to succeed in the course was measured separately from a student's motivation to go above and beyond the course expectations or excel beyond simply achieving success. The instrument did not provide a definition of 'success', leaving participants to interpret the meaning for themselves.

Operational Definitions of 'Motivation':

- Intrinsic Motivation
    i. "I am willing to participate and be successful in my learning within CS 102"

ii. "I desire to participate and be successful in my learning within CS 102"

iii. "I am willing to go above and beyond and do extra work to exceed course expectations for learning within CS 102"

iv. "I desire to go above and beyond and do extra work to exceed course expectations for learning within CS 102"

- Extrinsic Motivation

i. "I need to participate and be successful in my learning within CS 102"

ii. "I feel compelled (by someone else, like a parent) to participate and be successful in my learning within CS 102"

iii. "I need to go above and beyond and do extra work to exceed course expectations for learning within CS 102"

iv. "I feel compelled (by someone else, like a parent) to go above and beyond and do extra work to exceed course expectations for learning within CS 102"

- Motivation to Succeed

i. "I am willing to participate and be successful in my learning within CS 102"

ii. "I desire to participate and be successful in my learning within CS 102"

iii. "I need to participate and be successful in my learning within CS 102"

iv. "I feel compelled (by someone else, like a parent) to participate and be successful in my learning within CS 102"

- Motivation to Excel

i. "I am willing to go above and beyond and do extra work to exceed course expectations for learning within CS 102"

ii. "I desire to go above and beyond and do extra work to exceed course expectations for learning within CS 102"

iii. "I need to go above and beyond and do extra work to exceed course expectations for learning within CS 102"

iv. "I feel compelled (by someone else, like a parent) to go above and beyond and do extra work to exceed course expectations for learning within CS 102"

3.3.3 Intention to continue in Computer Science

It is not possible to measure or speculate about the effect of the game programming assignment on whether or not students currently enrolled in introductory computer science will remain in Computer Science through their degrees and into their careers, based on this study. The impact of the game programming assignment on student intention to continue in Computer Science can only be determined by an impact on a student's self-expressed *intentions* regarding the likelihood of continuing to take

Computer Science courses or programs. The study seeks to determine what changes occur to the students' long-term plans for Computer Science as a result of the game programming assignment. Students provided their intentions or plans relating to Computer Science immediately before and after the game programming assignment.

Operational Definition of 'Intention to continue in Computer Science'

i. "I am planning on taking more Computer Science courses"

ii. "I am planning on majoring in Computer Science"

3.3.4 Attitudes Towards Performance

To assess a learner's attitudes towards his or her own performance, a student's performance in the introductory course and the previous introductory Computer Science course and the student's perceived ability to succeed or perform in introductory Computer Science are assessed. A student's attitudes towards performance is referred to throughout the thesis and refers to both the student's measurable academic performance and self-measurement of ability to perform. Grades are indicators of performance and are included in this analysis because of their impact on attitudes towards performance.

Operational Definition of 'Attitudes towards performance':

- Performance
  - i. CS 101 Final Grade
  - ii. self-estimate of final mark in CS 102 near the end of the semester
- Perceived ability to perform

i. "I would rate my own ability to succeed in CS 102 as high"

Summary

The case study employed two questionnaires, one before the game programming assignment and one after the game program assignment, to investigate any differences in student learning during the game programming assignment in the introductory course. The exploratory questionnaires probed the four Constructs: student interest, motivation, intentions to remain in Computer Science and attitudes towards performance. An open-ended interview with the teaching assistant provided additional data to explore the impact of a game programming assignment on student interest, motivation, intentions to continue, and attitudes towards performance.

CHAPTER 4: ANALYSIS

4.1 Overview

The purpose of this analysis is to gain an understanding of the particular phenomena of the case study and the meaning for those involved. The limitations of an exploratory case study include generalizability beyond this single bounded system. The analysis is performed as an act of discovery and is not intended to provide a confirmation of any theory. The research analysis looks for patterns and trends that can inform future research and instructional practice.

Statistical analysis performed on the survey data includes descriptive statistics, comparison of means, variance analysis, discriminate analysis and correlation analysis.

4.2 Response Patterns

Immediately before students began the game programming experience, the pre-assignment questionnaire was administered. Students responded to questions using a Likert scale. The Likert scale responses were translated into numbers (strongly agree = 5; strongly disagree = 1). Twenty-four of the ninety-one students in CS102 for the 2005 / 2006 school year completed the first questionnaire. There was a three week interval between both questionnaires. Immediately after students completed the game programming experience, the post-assignment questionnaire was conducted with a response rate of twenty-eight of the ninety-one students. Thirteen students completed

both questionnaires. The raw questionnaire response data collected is also available in Appendix 3. The scores represent a convenience sample from the population under study.

4.3 Interest

Table 1: Paired sample t-test, interest (sample size = 13)

Item	Pre-Mean m(a)	Pre-SD SD(a)	Post-Mean m(b)	Post-SD SD(b)	Mean-Difference m(b-a)	t-test value (df)	p ($p<0.05$ statistically significant)
My impression of computer programming and problem solving is positive.	4.308	0.63	4.077	0.76	.231	t(12) = 1.897	.082
I would rate my interest in computer programming as high.	4.077	1.038	4.154	0.899	.077	t(12)=.562	.584
My impression of CS102 is positive.	2.231	0.599	2.923	1.115	.69231	t(12)=2.420	.032
I would rate my interest in CS102 as high.	2.846	1.068	3.846	1.068	1.000	t(12)=2.550	.025

Student interest in the introductory course and interest in the discipline of Computer showed a practically significant difference as shown by their average responses in Table 1: Paired Sample T-Test, Interest. Student interest in the discipline of Computer Science was high, with zero students reporting a low interest in the discipline. However, student interest in the course was lower than their interest in the discipline. The majority of

students were interested in the discipline of Computer Science (mean of 4.08) and *not* interested in the course (mean of 2.85).

Questions about student interest were reframed to assess the students' positive impression of the subject and course. This choice led to a challenge in interpreting the results without probing follow-up questions to understand any connection between a student's interest and his or her positive impression of something, like a course or a program of study.

By conducting a comparison of means and paired-sample t-test, we make the following observations:

1. There was not a statistically significant increase in the students' interest in, and positive impression of, the discipline of Computer Science. ($p>0.05$)
2. There was a statistically significant increase in the students' interest in, and positive impression of, the course from before and after the game programming assignment. (Mean of 2.84 increased to 3.84, $p<0.05$)

4.4 Motivation

Table 2: Paired sample t-test, motivation (sample size = 13)

Item	Pre-Mean m(a)	Pre-SD SD(a)	Post-Mean m(b)	Post-SD SD(b)	Mean-Difference m(b-a)	t-test value (df)	p ($p<0.05$ statistically significant)
I am willing to participate and be	3.4615	1.506	4.154	0.801	.692	t(12) = 1.897	.082

successful in my learning within CS102.							
I need to participate and be successful in my learning within CS102.	2.923	1.256	4	0.707	1.077	t(12)=3.742	.003
I desire to participate and be successful in my learning within CS102.	3.077	1.382	4	0.816	.923	t(12)=2.650	.021
I feel compelled (by someone else, like a parent) to participate and be successful in my learning within CS102.	2.154	1.067	2.917	1.311	.667	t(11)=2.345	.039
I am willing to go above and beyond and do extra work to exceed course expectations for learning within CS102.	3.154	1.345	3.462	1.391	.308	t(12)=1.171	.264
I need to go above and beyond and do extra work to	2.692	0.947	3.769	0.832	1.077	t(12)=3.742	.003

exceed course expectations for learning within CS102.							
I desire to go above and beyond and do extra work to exceed course expectations for learning within CS102	2.923	1.553	3.6615	0.961	.692	t(12)=1.996	.069
I feel compelled (by someone else, like a parent) to go above and beyond and do extra work to exceed course expectations for learning within CS102.	2	1.225	2.583	1.165	.500	t(11)=1.593	.139

Figure 1 shows the average measures of students' motivation before the game programming assignment in terms of the forms of motivation (intrinsic versus extrinsic) and succeeding or excelling. These distinctions in motivation, intrinsic and extrinsic, are defined and discussed in the Methodology chapter of this thesis.

Students reported higher intrinsic motivation than extrinsic motivation. Students reported being more motivated to succeed than to excel at their coursework. One possible explanation for this result is that students who are motivated to excel are implicitly also motivated to succeed since succeeding is a necessary pre-requisite for excelling.

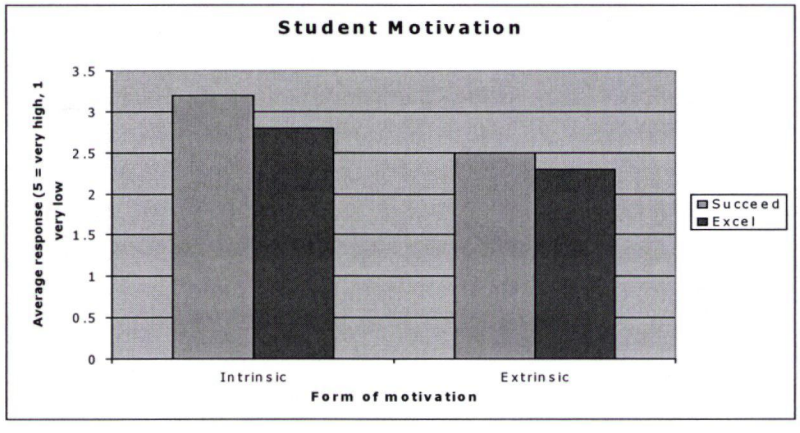

Figure 6: Student motivation before game programming assignment

By conducting paired-sample t-test, we make the following observations, as shown in Table 2:

1. There was a practically significant increase in reported motivation after the game programming assignment, both intrinsic (desire, willingness) and extrinsic (need, compelled). There was a statistically significant increase ($p<0.05$) in reported motivation for extrinsic motivation (need, compelled); however, there was not a statistically significant increase ($p<0.05$) in reported motivation for intrinsic motivation (desire, willingness).

2. There was a practically significant increase in reported motivation to excel or go above and beyond the course expectations after the game programming assignment. There was not a statistically significant increase ($p>0.05$).

4.5 Intent to Continue in Computer Science

Table 3: Paired sample t-test, intent to continue (sample size = 13)

Item	Pre-Mean m(a)	Pre-SD SD(a)	Post-Mean m(b)	Post-SD SD(b)	Mean-Difference m(b-a)	t-test value (df)	p ($p<0.05$ statistically significant)
I am planning on taking more Computer Science courses.	4.154	1.345	4.538	0.66	.385	t(12)=.1237	.240
I am planning on majoring in Computer Science.	3.462	1.506	4.154	0.899	.692	t(12)=2.420	.032

When students were asked before the game programming assignment if they intended to major in Computer Science, no students responded uncertainly: all students responded either "strongly agree" or "disagree". Only 40% of respondents in introductory Computer Science still intended to major in Computer Science. Interestingly, no female respondents indicated they would remain in Computer Science as majors nor take additional courses in Computer Science as non-majors. Due to this strong distinction between responses, two distinct groups of respondents are formed by this answer, for the purpose of comparison, based on their reported intentions to major in Computer Science: majors and non-majors.

By conducting a paired-sample t-test, we make the following observation about student intention to remain in Computer Science, as shown in Table 3:

1. Over the course of the game programming assignment, there was a statistically significant increase in the students' intention to claim Computer Science as their major. ($p<0.05$)
2. There was not a statistically significant increase in the students' intention to take more courses in Computer Science. ($p>0.05$)

4.6 Attitudes Towards Performance

Table 4: Paired sample t-test, attitudes towards performance

Item	Pre-Mean m(a)	Pre-SD SD(a)	Post-Mean m(b)	Post-SD SD(b)	Mean-Difference m(b-a)	t-test value (df)	p ($p<0.05$ statistically significant)
What was your final mark in CS101?	3.962	1.163					
What is your estimated final mark in CS102?			3.667	1.435			
I would rate my ability to succeed in CS102 as high	3.462	1.33	3.846	1.214	.385	t(12)= 1.594	.137

In order to determine a student's belief about his or her ability to succeed in Computer Science, past performance was measured (final grade in CS 101). They were also asked to rate their own ability to succeed in the course (CS 102).

By conducting a comparison of means and paired-sample t-test, the following observation can be made about student attitudes towards performance, as shown in Table 4:

1. Using comparison of means, there was an increase of 7.7% in students' self-efficacy or belief in their own ability to succeed in Computer Science. Although practically significant, this was not a statistically significant difference. ($p>0.05$)

4.7 Comparison by Groups

For analysis purposes, students were separated into distinct groups. This comparison by groups using discriminate analysis may provide a clear profile for the purpose of classifying students into groups. These classifications, while not allowing predictions, can be useful to identify relationships between the collected data: are the set of answers different from one group to the next?

Gamer Vs. Non-Gamer

Students were separated into two groups to explore a possible difference attributable to game experience:

1. Students who reportedly enjoyed and frequently played video games
2. Students who did not enjoy or frequently play video games.

This comparison by groups using discriminate analysis could not provide clear distinctions. Overall, the data for both groups was similar with the following distinctions:

1. Interest in the course increased after the game programming assignment for non-gamers more so than gamers. (Non-gamer means: Pre of 2.80 to Post of 3.88; Gamer means: Pre of 3.00 to Post of 2.86)
2. Non-gamers reported higher intrinsic motivation than gamers to succeed in the course. (Non-gamer means: Pre of 3.35 to Post of 4.03; Gamer means: Pre of 2.75 to Post of 3.29)
3. Non-gamers reported an increase in self-efficacy; whereas, gamers reported an average decrease in perceptions of their ability. (Non-gamer means: Pre of 3.50 to Post of 4.13; Gamer means: Pre of 4.00 to Post of 3.57)

This information indicates that game programming assignments do not present a disadvantage to non-gamers. Any benefit that might be attributed to the game programming assignment, through an increase to either interest, motivation, intent to remain in Computer Science or attitudes towards ability, is not limited to gamers. In fact, there was a more significant increase on the four constructs for non-gamers than gamers. This was contrary to the research expectations that the game programming assignment would present an advantage to gamers.

Procrastination

In an attempt to identify if students procrastinated less when facing a game programming assignment, 87% of respondents reportedly spent more time on Assignment 5 than any previous assignment in the course. Possible explanations of this result may include that this was the last, and largest, assignment of the semester. The questionnaire did not

properly identify whether or not students started the assignment earlier (more days before the deadline) rather than waiting until the "last minute". The questions are based on self-reported time spent on the assignment, and thus need to be interpreted cautiously as these may over or under report actual time spent on the assignment.

Significant Differences between Majors and Non-Majors

Due to the clear distinction between students who plan to major in Computer Science (40%) and students who plan to not major in Computer Science (60%), two groups are formed for the purpose of analysis. Student interest appeared to have a weaker correlation with student motivation and success for students who were not intending to major in Computer Science. Both Majors and Non-Majors played a similar amount of games and had a similar level of motivation to excel. However, non-majors were statistically more likely to prefer non-game programming assignments, reported lower interest in the discipline, lower interest in the course, and slightly higher extrinsic motivation than majors. Non-majors, through comparison of means, also reported lower perceived success and lower grades. Conversely, majors reported higher interest in the discipline and interest in the course.

There was a larger increase in the interest and motivation of majors than non-majors after the game programming assignment. Since there was a strong increase in students intending to major in Computer Science (comparison of means result of 13.8%), the students who became interested *after* the game programming assignment are of particular

interest: their interest in the course and the student motivation, both intrinsic and extrinsic, increased.

4.8 Relationships Between Constructs

In order to determine whether any relationships existed between our four constructs, Pearson's correlation analysis was undertaken between each question on the instrument. These questions were then grouped into four constructs: student interest, student motivation, intention to remain in Computer Science, and attitudes towards performance. For the purpose of this study, it was assumed that the constructs are independent, normally distributed, continuous and that there can be a linear relationship between the two constructs compared. Where a correlation is stated to be strong, the $p<0.05$. The raw correlation matrix is found in Appendix 2.

There was a strong correlation between student interest in the discipline of Computer Science and intrinsic motivation. This was a statistically significant result indicating that student motivation to succeed and perform was related to how interested he or she was in the discipline. These students who were interested in the discipline of Computer Science were also more likely to report an intention to major in Computer Science and reported higher previous success. There was also a strong correlation between student interest in the course, considered separately from the discipline of Computer Science, and motivation to succeed, both intrinsically and extrinsically.

Motivated students (students most motivated to succeed) reported higher interest in Computer Science and the introductory course. Motivated students also reported better past performance in Computer Science than students who reported lower motivation. Contrary to researcher expectations, student motivation to excel was not significantly correlated with interest, retention, or attitudes towards performance.

Strong correlations indicated that if students were not interested in the discipline of Computer Science, then they were less likely to major in Computer Science. There was a strong correlation between past successes in Computer Science and intent to remain in Computer Science. Despite the strong correlation between past success, no significant correction was determined between a student's perceived ability to succeed and his or her motivation, interest, retention, or even past success. This was also contrary to researcher expectation.

It was expected by the researcher that students who are more interested experience greater success. Student interest in the course was statistically significantly higher after the game programming assignment. Strong correlations (where $p<0.05$) suggest that students who are more interested, both in the course and in the discipline, appear to be more intrinsically motivated, and more likely to remain in Computer Science. Students who are more interested in the discipline appear to have a higher self-efficacy.

In an interview with the teaching assistant responsible for marking all assignments and assisting students throughout the course of the assignment, Joey[1] provided his own observations. The interview was an informal, unstructured interview of 20 to 30 minutes based on some samples of student work. Joey observed that students were *much* more highly motivated to complete the game programming assignment than the last four assignments of the semester. Joey felt that the students who went "above and beyond" and completed a superior game were the most satisfied with their experience. While marking student assignments, he was very pleased with the quality of assignment including some students including full graphics and very creative collision detection. Joey was surprised and impressed with the effect on student participation, motivation, and performance of the game programming assignment and said that this effect was independent of the students' choice of which game to complete. Joey's observations appear to be echoed by our analysis of survey data.

Summary

The purpose of this analysis is to gain an understanding of the particular phenomena of interest in the case study and the meaning for those involved. Questionnaires and an open-ended interview were used to explore the relationship between student interest, motivation, intentions to remain in Computer Science and attitudes towards performance, and a game programming assignment. An exploratory analysis performed on the

---

[1] Pseudonym

questionnaire data included descriptive statistics, a comparison of means, variance analysis, discriminate analysis and correlation analysis.

CHAPTER 5: DISCUSSION AND CONCLUSION

5.1 Overview

This case study provided the opportunity for empirical inquiry into a contemporary phenomenon, game programming in a computer science course. The phenomenon, the relationship between game development assignments and student learning, motivation, interest and self-efficacy, was investigated within its own real-life context: the post-secondary classroom. The findings from this case study increase our understanding of the relationship between a game programming assignment and aspects of the learning experience of first-year post-secondary computer science students. Although not generalizable, the findings from this study are promising and suggest some promising areas of future research and teaching practice. In this chapter I will highlight and discuss key findings, address limitations of the study, and make recommendations for future research.

5.2 Results

The key findings in this case study come from the analysis of two questionnaires and a brief interview. Immediately before students began the game programming experience, the pre-assignment questionnaire was conducted. There was a three week interval between both questionnaires. Immediately after students completed the game

programming experience, the post-assignment questionnaire was conducted. 13 students completed both questionnaires.

Students before and after the game programming assignment reported a change in some key factors associated with learning: student interest, motivation, intentions to remain in Computer Science and attitudes towards performance. A promising finding of this research is that student interest in the introductory Computer Science course increased from before to after the game programming assignment experience. This finding was not accompanied by a statistically significant increase in student interest in the discipline of Computer Science. One possible explanation for this difference is that the student interest in the discipline (mean = 4.15) was initially higher than student interest in the course (mean = 2.85) and thus there was different opportunity for increase. Another possible explanation for this finding is that the learners related or attributed their learning experience more to the course than to the discipline of Computer Science. While the game programming assignment could be a leading contributor to the statistically significant increase in student interest in the course, other possible explanations could include changes in instruction or a new topic in lecture, which should be investigated in a future study.

The results also indicated a practically significant increase in reported motivation and attitudes towards the learners' own ability to succeed, or self-efficacy. Similar to the increase in student interest, while the game programming assignment could be a leading contributor to the practically significant increase in student motivation and self-efficacy

in the course, other possible explanations could include changes in instruction or a new topic in lecture. Another possible explanation that should be considered as having a possible effect on student motivation would be the lapse of three weeks between the questionnaires leading the learners closer to final semester deadlines and examinations. While there are alternate possibilities, it is a promising finding that student motivation and self-efficacy appeared to increase.

The study found a statistically significant increase in the students' reported intention to major in Computer Science. This is a promising finding, and one that was unexpected by the researcher. Students reported that they were more likely to major in Computer Science after the game programming assignment was completed. While an increase in student interest or motivation may be expected to lead to an increase in student intention to major in Computer Science, it was a hopeful finding to see a statistically significant increase ($p < 0.03$). As described in Section 2.2 of the Literature Review, a common concern facing Computer Science programs across North America is the low rate at which Computer Science students are graduating. The potential link between game programming assignments and intent to major in Computer Science should be studied further. As discussed in the Analysis Chapter, enrolment in Computer Science follows a cycle of increases and decreases historically related to local economy. The difference in reported plans from the pre-assignment questionnaire to the post-assignment questionnaire occurred over a short period of time. This finding is deserving of further study into the possibility that game programming assignments may affect a student's plan to major in Computer Science. It would also be worth further study to explore reasons

for any increase in retention, such as a link with increased interest or motivation. These statistical results are detailed in the Analysis Chapter.

During an interview, the teaching assistant of the studied course observed that students appeared to be more highly motivated to complete the game programming assignment than the previous four assignments of the semester. The teaching assistant observed that the students who went "above and beyond" and completed a superior game appeared to be the most satisfied with their experience in the course. While marking student assignments, the teaching assistant was pleased with the quality of assignments completed by students, some of whom included full graphics and very creative collision detection. The teaching assistant reported feeling surprised and impressed with an observed increase in student participation, motivation, and performance and said that these behaviours appeared to be independent of the students' choice of which game to complete. These interview comments echo the significant findings of the study's analysis. Based on the observations reported by the teaching assistant it is recommended to design interesting assignments to engage learners in complex learning tasks. It is also recommended that opportunities are provided for teaching assistants or instructors to reflect on their teaching practices and outcomes. The interview with the teaching assistant yielded promising feedback about what the assignment meant to the students and to the instructor; in a future study, more focused and detailed questions might be drafted to probe the teaching perspective on game programming assignments in more detail.

5.3 Limitations of the Study and Recommendations for Further Research

This was a case study of the relationship between a game programming assignment and key aspects of successful student learning. The sample who provided data was small, and the data collection focused on enrolees in one computer science course. While game programming assignments appear to be related to increased interest and motivation, the use of this teaching approach should be studied more in the future with a larger sample and across several courses. The results of this case study cannot be causally attributed to the game programming assignment due to the study design. While establishing causality is a common research goal in the physical sciences, causal relationships are not as easy to attain across the social sciences given the uncontrollable variables surrounding research with human subjects. To establish a causal relationship between the game programming assignment and effects on student learning, future research provides the opportunity to establish strong internal validity by performing pattern-matching, explanation-building and addressing rival explanations. To generalize the results beyond this classroom, future research may define a domain to which this can be generalized with external validity, such as an experimental or quasi-experimental design (Yin, 2003).

This case study employing questionnaires was designed as an exploratory analysis to generate hypotheses worthy of further study or investigation:

- if learners complete a game programming assignment, does this lead to greater engagement with their learning?

- if learners complete a game programming assignment, does this leads to a more positive perception of students' own performance
- if learners complete a game programming assignment, does this affect students' intentions to major in Computer Science?

Further study into the hypotheses generated by this research would be enabled by conducting a similar study that addresses the limitations of this case study. In order to carry out a study that would yield more generalizable results, these aspects of the study design could be improved, addressing the limitations of this study:

- modifying the survey to use pre-defined operational measures for the educational research concepts like 'motivation' and 'self-efficacy', providing stronger construct-validity;
- using an experimental control group which may complete an assignment which is not a game-programming assignment but covers similar learning objectives; and,
- broadening the study across a larger population and sample size perhaps covering more than one section of the introductory course or conducting the study at multiple universities.

With modifications or improvements in place, a similar study could be conducted from which recommendations for Computer Science teaching practice could be derived. For example, this researcher recommends a case study across two sections of the same course taught by the same instructor but offering different assignments.

The way in which questions were worded in the questionnaire made the interpretation of results difficult and often led the researcher to consider probing follow-up questions. One problematic phrase was 'compelled by parents' as an indicator for extrinsic motivation which seemed confusing upon review. Another example was reframing questions about a learner's interest as how positive was their overall impression of the subject matter or course. It's not clear that a positive impression is similar enough to having an interest. Pre-defined operational measures for interest, motivation and self-efficacy would aid in the interpretation of results.

5.4 Discussion

This exploratory case study provided an opportunity to better understand the relationship between game development assignments and student learning, motivation, interest and attitudes towards performance in a computer science course. Several research questioned guided the study, and can be turned into hypotheses worthy of further study or investigation. Promising findings included reported increases in student interest, motivation, self-efficacy and intention to major in computer science after completing a game development assignment. The results of this case study appear to indicate that game programming assignments can positively impact aspects of student learning and are worthy of further study.

An exemplary case study, according to Yin (2003) and as described in Section 2.5 of the Literature Review, must be significant, complete, consider alternative perspectives,

display sufficient evidence and be composed in an engaging manner. The significance of this case study, that the concepts being studied are worthy of study, is established in Sections 2.2, 2.3 and 2.4 of the Literature Review. Multiple sources of evidence were analysed to assure the completeness of the study, namely the questionnaires and the interview. Alternative perspectives were considered and addressed as part of the analysis and discussion of the results. The sufficiency of evidence displayed as part of this case study could be improved by performing a similar study with the above-noted improvements.

An area that may warrant future study is the relationship between gender and success in computer science. In the present study, gender was intentionally excluded from the analysis. This is because the small number of female participants did not provide a reasonable sample size for analysis. However, there may be a difference in how male and female computer science students participate in learning assignments and is worth studying further with a larger population.

The benefits to students of a game programming assignment in terms of career-specific training and experience for the game industry was intentionally excluded. The growing computer game industry is hiring Computer Science graduates and requires applicants who are experienced and trained in game programming. This is intentionally left out from the study in order to concentrate on benefits to student learning, rather than the course being part of a training program for game programmers or non-game programmers.

A researcher expectation was that the game programming assignment may impact learners who were gamers differently from learners who were non-gamers. Wilson (2002) found that game playing was negatively predictive of success in computer science, meaning that students who played video games the most, were generally the least successful at learning computer science. In this study, non-gamer participants self-reported a practically higher intrinsic motivation and interest than gamers. Non-gamer study participants reported a practically significant increase in self-efficacy; whereas, gamers reported an average decrease in perceptions of their ability. Non-gamers did not report a greater change before and after the game programming assignment than gamers.

5.5 Conclusion

This chapter addressed key findings and limitations and makes recommendations for future research as a result of the study. The relationship between a game programming assignment in Introductory Computer Science and student motivation, interest, intention to remain in Computer Science and self-efficacy was analysed through a case study. The results of this empirical inquiry demonstrated a promising increase in student interest in the course, motivation, self-efficacy and intention to major in computer science. These results encourage further study into the use of game programming assignments by

Computer Science educators. Although not generalizable, the findings from this study are promising and suggest some promising areas of future research and teaching practice.

The research aim of exploring the effect of game programming assignments on student learning provided exploratory results with promising findings to encourage further research into the topic. Limitations in the study design suggest that the findings be interpreted conservatively and with an understanding that more research into game programming assignments is needed to build upon and extend these results.

REFERENCES

Alexander, S., Clark, M., Loose, K., Amillo, J., Daniels, M., Boyle, R., Laxer, C., and Shinners-Kennedy, D. 2003. Case studies in admissions to and early performance in computer science degrees. *SIGCSE Bull. 35,* 137-147. http://doi.acm.org/10.1145/960492.960541

Astin, A.W., and Oseguera, L, 2005. *Degree attainment rates at American colleges and universities.* Higher Education Research Institute, Los Angeles, CA.

Atari, 1979. Asteroids [Computer Software]

Atari, 1980. Centipede [Computer Software]

Bandura, A. 1977. Self-efficacy: Toward a unifying theory of behavioral change. *Psychological Review 84*, 191-215.

Bandura, A. 1986. *Social foundations of thought and action: a social cognitive theory.* Prentice Hall, Englewood Cliffs, N.J.

Beaubouef, T., and Mason, J., 2005. Why the high attrition rates for computer science students: some thoughts and observations. *inroads - The SIGCSE Bulletin 37*, 103-106.

Becker, K., and Parker, J., 2005. All I ever needed to know about programming, I learned from rewriting classic arcade games. In *Proceedings of Future Play Conference*,(East Lansing, MI) Michigan State University, 2005.

Becker, K. 2001. Teaching with games: the Minesweeper and Asteroids experience. *Journal for Computing in Small Colleges 17,* 23-33.

Bergin, S., Reilly, R. 2005. The influence of motivation and comfort-level on learning to program. In P. Romero, J. Good, E. Acosta Chaparro & S. Bryant (Eds). *Proc. PPIG 17,* 293-304.

Connolly, R., 2004. A funny thing happened on the way to the form: using game development and web services in an emerging technology course. *Information Systems Education Journal, 3.* No. 38. http://isedj.org/3/38

DeLaet, M., Slattery, M.C., Kuffner, K. and Sweedyk, E., 2005. Computer games and CS education: Why and how. In *Proceedings of the 36th SIGCSE technical symposium on Computer science education,* (St. Louis, Missouri, USA, February 23 - 27, 2005). SIGCSE '05. ACM, New York, NY, 256-257.

Egenfeldt-Nielsen, S., 2006. Overview of research on the educational use of video games. *Digital Kompetanse 1* 184-213.

Eysenck, H. J., 1976. Introduction. In H.J. Eysenck (Ed.)., *Case studies in behavior therapy*. Routledge, London, UK.

Flyvbjerg, B., 2006. Five Misunderstandings about case-study research. *Qualitative Inquiry 12.*

Haden, P., 2006. The incredible rainbow spitting chicken: teaching traditional programming skills through games programming. In *Proceedings of the 8th Australian Conference on Computing Education 52* . D. Tolhurst and S. Mann, Eds. Australian Computer Society, Darlinghurst, Australia, 81-89.

Jones, R., 2000. Design and implementation of computer games: a capstone course for undergraduate computer science education. In *Proceedings of the 31st SIGCSE Technical Symposium on Computer Science Education 32.*

Kafai, Y., 1995. *Minds in play: computer game design as a context for children's learning*. Mahwah, N.J.

Konami, 1981. Frogger [Computer Software]

Ladd, B., 2006. The curse of Monkey Island: holding the attention of students weaned on computer games. *Journal of Computing in Small Colleges 21,* 162-174.

Lawrence, R., 2004. Teaching data structures using competitive games. *IEEE Transactions on Education 47,* 459-466.

Leska, C. and Rabung, J. 2005. Refactoring the CS1 course. *Journal of Computing in Small Colleges 3,* 6-18.

Leutenegger, S. and Edgington, J. 2007. A games first approach to teaching introductory programming. *SIGCSE Bull. 39 1,* 115-118. http://doi.acm.org/10.1145/1227504.1227352

Lobo, A. F., Baliga, G. R., Bergmann, S., Stone, D., and Shah, A. 2000. Using real-world objects to motivate OOP in a CS1 lab. In *Proceedings of the Fifth Annual CCSC Northeastern Conference on the Journal of Computing in Small Colleges* (Ramapo College of New Jersey, Mahwah, New Jersey, United States). J. G. Meinke, Ed. Consortium for Computing Sciences in Colleges, 144-156.

Lomerson, W., and Pollacia, L., 2006. Declining CIS enrolment: an examination of pre-college factors. *Information Systems Education Journal 4,* No. 35.

Macedonia, M., 2002. Games, simulations, and the military education dilemma. In *Internet and the University: 2001 Forum,* M. Devlin, R. Larson, & J. Meyerson, Eds. Cambridge, MA, 157-167.

Malone, T. W. 1980. What makes things fun to learn? heuristics for designing instructional computer games. In *Proceedings of the 3rd ACM SIGSMALL Symposium and the First SIGPC Symposium on Small Systems*, New York, NY, 162-169. http://doi.acm.org/10.1145/800088.802839

Mitchell, M., Sheard, J., and Markham, S. 2000. Student motivation and positive impressions of computing subjects. In *Proceedings of the Australasian Conference on Computing Education* (Melbourne, Australia). A. E. Ellis, Ed. ACSE '00, vol. 8. ACM, New York, NY, 189-194. http://doi.acm.org/10.1145/359369.359398

Papert, S., 1980. *Mindstorms: Children, Computers and Powerful Ideas.* Basic Books, Inc, New York.

Papert, S., 1996. *The connected family: bridging the digital generation gap.* Longstreet Press, Marietta, GA.

Parberry, I., Roden, T., and Kazemzadeh, M., 2005. Experience with an industry-driven capstone course on game programming: extended abstract. In *Proceedings of the 36th SIGCSE Technical Symposium on Computer Science Education*, New York, NY, 91-95. http://doi.acm.org/10.1145/1047344.1047387

Piaget, J., 1950. *The Psychology of Intelligence.* Routledge, New York.

Prensky, M., 2001. Digital Natives, Digital Immigrants. *On the Horizon 9*, No. 5.

Reversa, Reversa Website. http://www.seemoresideeffects.ca Last accessed Feb 6, 2008.

Roberts, E., 2000. Strategies for encouraging individual achievement in introductory computer science courses. *SIGCSE Bulletin 32*, 295-299. http://doi.acm.org/10.1145/331795.331873

Rosas, R., Nussbaum, M., Cumsille, P., Marianov, V., Correa, M., Flores, P., Grau, V., Lagos, F., López, X., López, V., Rodriguez, P., and Salinas, M. 2003. Beyond Nintendo: design and assessment of educational video games for first and second grade students. *Computing Education 40,* 71-94. http://dx.doi.org/10.1016/S0360-1315(02)00099-4

Sindre, G., Line, S., and Valvåg, O. V. 2003. Positive experiences with an open project assignment in an introductory programming course. In *Proceedings of the 25th international Conference on Software Engineering* (Portland, Oregon, May 03 - 10, 2003). International Conference on Software Engineering. IEEE Computer Society, Washington, DC, 608-613.

Taito, 1978. Space Invaders [Computer Software]

Taulbee Survey, Computing Research Association. Retrieved February 8, 2008 from http://www.cra.org/statistics

Fact Books 1998 – 2007, University office of institutional analysis. Retrieved May 15, 2009, from University of Calgary Web site: http://www.ucalgary.ca

Valentine, D. 2005. Playing around in the CS curriculum: reversi as a teaching tool. *Journal of Computing in Small Colleges 20*, 214-222.

Vedrashko, I., 2006. *Advertising in computer games.* Thesis, Massachusetts Institute of Technology. Retrieved Jul 27, 2009, from http://dspace.mit.edu/bitstream/handle/1721.1/39144/123290221.pdf?sequence=1

Vegso, J, 2005. Interest in CS s a major drops among incoming freshmen. *Computing Research News 17.*

Vegso, J., 2007. Continued Drop in CS Bachelor's Degree Production and Enrollments as the Number of New Majors Stabilizes. *Computing Research News 19*, No. 2

Wilson, B. C. and Shrock, S. 2001. Contributing to success in an introductory computer science course: a study of twelve factors. *SIGCSE Bulletin 33,* 184-188. http://doi.acm.org/10.1145/366413.364581

Wilson, B., 2002. A study of factors promoting success in computer science including gender differences. *Computer Science Education 12,* 141-159.

Yin, R., 2003. *Case study research: design and methods 3rd edition.* Sage Publications.

Student:_____ Asst:_____ Page 1

General Programming Assignment Rubric

Attribute	Unacceptable	Meets Requirements	Exemplary		Points
Presentation					
Information	**Missing Information**	**Properly prepared for submission;** includes all names; TA; Lab section; Assignment number; Statement of Originality	Nothing missing – purpose & program highlights obvious.		
Files	**Missing files**	**All included:** required source, data, documentation, script, instruction files	Nothing missing – clear labeling of all parts; readme file includes everything needed to run & test this program		
Submission:	**> 3 min.** :Unable to figure out how to compile & run after 3 minutes.	**2-3 min.** Takes 2-3 minutes to figure out how to compile & run.	**< 2 min.** Takes less than 2 Minutes to compile; run; test		
Demo:	**Major Problems:** Program won't run or crashes unexpectedly. Submitted code seems different from that demonstrated. Programmers seemed confused by the program they were demonstrating. Only one group member participated (out of three).	**Minor Problems:** Program runs reasonably well. Knew what they were doing. Submitted code matches demo. One group member did not participate.	**Good!** Demo went well; major features shown and explained. Demonstrators clearly understood their work. All group members participated.		
				Presentation Total	/6
Documentation					
Overall Impression	**Misleading**; confusing; too much or too little	A **few** missing, redundant, or irrelevant parts	**Accurate**; reasonable; easy to read;		
Identifier Names	**Meaningless** or misleading names	**Some** poor choices. Most identifiers explained where appropriate.	**Meaningful** identifier names [some single letter names are OK, such as i,j for indices]. Explanations of identifiers where appropriate.		
Indentation; White Space	**Misleading** indentation; too much or too little white space	**Some** inconsistencies; some inadequate or wasted space	**Consistent** indentation; good use of white space		
External Documentation	**No** external documentation when some is needed. External documentation not useful, confusing, out of date, or misleading	**Adequate** external documentation [javadoc; user manual, as necessary]. Someone else could work with this program with a little help from the original programmer(s).	External doc. as **appropriate**; [javadoc & user documentation where appropriate]. Someone else could work with this program based on code and documentation alone.		
Logical Blocks	**Few** or no logical blocks documented	**Most** logical blocks documented	Documentation for **each** function and loop and logical block		
When Defining Classes:	**Missing** class diagrams	**Reasonable** attempt at class diagrams	Includes some readable form of **class diagram**		
Single Static Class or "C"	**Missing** flowchart	**Reasonable** attempt	Flowchart or other **readable representation** of program flow		
				Documentation Total	/12
Design					
Implemented what was asked.	**Hardly** followed specs; no explanations for deviations	Some **deviations**	**Followed specs** (deviations well justified)		
Style & Efficiency	**Poor** choices of code; awkward structures	**Mostly well thought out;** most parts reasonably efficient	**Well thought out;** reasonably efficient (code & data structures)		

Program Subdivisions	**Too many or too few**	**Mostly reasonable** with a few poor choices	**Reasonable** subdivisions (classes/ functions)		
Flow	**Confusing**; hard to follow	**Some awkward** or confusing parts	**Logical**, justifiable structure		
Constants; Magic Numbers	**Magic numbers**; hard-coded values	**Most** constants named & explained	**Appropriate** use of constants		
Globals	**Inappropriate** use of globals.	**Most** globals properly justified.	**All** globals properly justified.		
Initialization & Clean-up	**Uninitialized** values; no clean-up	**Some** uninitialized values, some clean-up **missing**.	**Good!** Reasonable initialization and clean-up		
Structure	**Not clear –** blocks do too much or too little	**Reasonable,** given degree of difficulty of assignment and learner's expected level of experience	**Excellent:** creative, logical, well-delineated with respect to tasks and data		
			Design Total		/16
Input / Output / Interface					
Intro / Finish	**No** header or introduction; no obvious end	**Has start & finish.**	**Good!** Has nice introduction; clear "sign-off";		
Output (or public interface for Class):	Output **unexplained**.	Output **readable**.	**Good!** Output is clean and reasonable.		
Input : Interactive Programs:	**No explanation** of input requirements	**Minimal** explanation of input requirements	**Good!** Input requirements explained as program runs		
Input : "Batch"(or 'user' interface for Class):			Input requirements **well explained** in external documentation		
			Input / Output / Interface Total		/6
Testing / Error Detection / Correction					
Choice of test data	**Missing** test data; poorly tested	Only **tested with given data** or only partially tested	**Thorough**; includes some data not given		
Annotation	**Not annotated;** hard to find & follow the testing; confusing	**Reasonably well annotated;** fairly easy to see & follow the testing	**Well annotated**; easy to see & follow the testing		
Endpoints	**Insufficient**; Only one or two endpoints tested	**Most** endpoints tested	**All** endpoints tested as appropriate		
Detection; Correction; Limitations	**Non-existent** error detection/ no attempt at correction/ unreasonable limitations	**Mostly reasonable** error detection/ attempted correction/ mostly reasonable limitations	**Reasonable** error detection/ correction/ limitations		
Debugging Aids	**No** evidence of debugging aids	**Some** evidence of built-in debugging aids	**Reasonable** use of DEBUG flags, program tracing and other debugging aids		
			Testing / Error Detection / Correction Total		/10
			[Rubric One] Style & Design Total:		/50

CPSC 233 Programming Assignment Rubric
Assignment 5 - Advanced:
Inheritance & Polymorphism

+Please report any omissions, errors, inconsistencies, or misleading explanations so they can be clarified.

Brief problem statement:

This assignment involves creating an application in Java that:

- includes the use of several separately compiled classes
  - demonstrates the use of inheritance and polymorphism
-

These classes should incorporate the use of inheritance and polymorphism.

Examples include, but are not limited to:

- **Library Utility (including a main program whose purpose is to demonstrate the utility): Linked List Library, Matrix Library, Drawing Utility**
- **Games: Asteroids, Centipede, Frogger, Space Invaders (games will likely need to be demo'd to ensure adequate assessment)**

The use of PACKAGES is still not required, and will not be considered an asset in this application.
The following requirements apply to all fifth assignments.
If you choose to submit a program that does not have a grading scheme described, the table below lists the elements that must be demonstrated.

This fourth assignment will be assessed as 'A', 'B', 'C', 'D' or 'F', with possible pluses or minuses.
This rubric is a guide: no further detailed description will be provided for this portion.
This rubric is meant to be used in conjunction with the general programming rubric.

Attribute	Attempt	Meets Requirements	Exceeds Requirements	Exemplary
Classes and subclasses	Still only one class, or more than one class, but all still static.	Reasonably clear, relatively independent classes.	Classes are neither too big nor too small; can be logically justified.	
Inheritance: creation & use	---	effective instantiation & use of pre-defined derived classes	able to define and create simple inherited classes	consistent, effective use
Class hierarchies	Crude outline.	Can create accurate representation of the class hierarchy in the program.	Uses UML-like notation to distinguish relationships between classes and to highlight attributes and behaviours.	Uses UML notation to distinguish relationships between classes and to highlight attributes and behaviours.
Overriding / Overloading	One example but not	1 -2 examples, moderately well justified. Code shows distinction between	Able to use appropriately, justified in	Clean, obvious, well-justified.

	justified.	the purposes of overloading/overriding.	documentation.	
Polymorphism / Abstract classes	Code compiles.	Uses at least one base class and two derived classes.	Appropriate use. Defines abstract classes as appropriate.	Consistent, well-justified.
Exceptions	Can follow valid code. Able to use simple ones in code copied from an example.	Can use necessary constructs. Can create simple exception handlers.	Documentation clearly indicates use is conscious and deliberate. Can propagate one exception through multiple levels of code.	Able to detect and correct subtle errors in code examples. Organizes code to avoid frailty. Can propagate multiple exceptions through multiple levels of code.
Ability to use pre-defined exceptions.	Hand execution of valid code.	Can use pre-defined exceptions.	Recognize different classifications of exceptions.	Clear understanding of appropriate use of exceptions.
Ability to define new exceptions.	Hand execution of valid code.	Can create simple exceptions.	Create exceptions that recover conditions.	Well-defined; justified.
Describe how Event-Driven Programs differ from Data-Driven programs.	Describe, in general terms.	Give an example of each.	Give several distinct examples where each approach would be appropriate.	Able to describe in general terms, the kinds of problems that are appropriate to each.
Understand the Java Event Model	Sort-of.	Describe.	Describe with visual examples. Distinguish between an event and an exception.	Distinguish between an event and an exception. Implement.
Implement event-handling methods.	Describe, generally. Can follow valid code.	One working example. Can create simple event handlers.	Two different examples. One coded. Can propagate one event through multiple levels of code.	> 1 different coded examples. Can propagate multiple events through multiple levels of code.

Points: General Grading Explanation:

A	4.0	Exemplary	goes well beyond the requirements as laid out in the assignment specifications.
B	3.0	Exceeds requirements	goes beyond the basic requirements as laid out in the assignment specifications in ways that add value and meaning to the solution within the context of the objectives for this course.
C	2.0	Meets requirements	as laid out in the assignment specifications.
D	1.0	Attempt	submission suffers from serious or un-ignorable flaws or difficulties.
F	0.0	Fail	

Assessment: Letter Grade Mappings				
A = Exemplary	B = Excedes Minimal	C = Meets Minimal	D = Sub-Standard	F = insufficient

			Requirements			Requirements						
	A	A-	B+	B	B-	C+	C	C-	D+	D	F (reasonable attempt)	not submitted
GPA	4.0	3.7	3.3	3.0	2.7	2.3	2.0	1.7	1.3	1.0	0.5	0.0
/50	50	46	42	38	34	30	25	22	17	13	6	0
/100	100	92	84	76	68	60	50	44	34	26	12	0

2: Raw correlation matrix

	IA.2	IB.1	IB.2	IC.1	IC.2	ID.1	ID.2	MA.1	MA.2	MB.1	MB.2	MC.1	MC.2	MD.1	MD.2	ME.1	ME.2	MF.1	MF.2
IA.1	.817(**)	.725(**)	.792(**)	.679(*)	.629(*)	0.2	.695(**)	0.54	.559(*)	.559(*)	0.187	.736(**)	0.486	-0.447	-.641(*)	0.333	0.205	0.172	0.147
	0.001	0.005	0 001	0 011	0.021	0 513	0 008	0.057	0 047	0 047	0.541	0 004	0 092	0 125	0 025	0.267	0.502	0.575	0 633
	13	13	13	13	13	13	13	13	13	13	13	13	13	13	12	13	13	13	13
IA.2		.732(**)	714(**)	0.507	0.303	0 427	0.529	0.549	0.527	0.444	0.31	867(**)	0 403	-0 324	- 657(*)	0.477	0.279	0.151	0.162
		0.004	0.006	0.077	0.315	0 146	0.063	0.052	0.064	0.129	0.302	0	0 172	0.28	0.02	0.099	0.356	0.621	0.596
		13	13	13	13	13	13	13	13	13	13	13	13	13	12	13	13	13	13
IB.1			880(**)	0 371	.582(*)	0 312	0 237	0.349	586(*)	0 197	0.114	635(*)	0 393	-0 312	-0 465	0 289	0.089	0.196	0 022
			0	0.212	0.037	0.299	0.435	0.243	0.035	0.519	0.712	0.02	0.184	0.299	0.128	0.337	0.773	0.522	0.942
			13	13	13	13	13	13	13	13	13	13	13	13	12	13	13	13	13
IB.2				0.393	0.512	0.2	0.461	0.497	.659(*)	0.233	0	.661(*)	0 454	-0 461	-0 456	0.255	0.072	0.06	-0.06
				0.184	0.074	0.512	0.113	0.084	0.014	0.444	1	0.014	0.119	0.113	0.137	0.401	0.816	0.845	0.846
				13	13	13	13	13	13	13	13	13	13	13	12	13	13	13	13
IC.1					0.403	0.06	0.321	0.241	-0.08	0.469	0.393	0.379	0	-0.321	-0.53	-0.048	-0.238	-0.158	0.283
					0.172	0 845	0.286	0.427	0.795	0.106	0.184	0.201	1	0 286	0 076	0.877	0.433	0.606	0.349
					13	13	13	13	13	13	13	13	13	13	12	13	13	13	13
IC.2						0.059	0.339	0.469	0.388	0.233	-0.211	0.274	0.366	-0.059	-0.184	0.064	-0.083	0.212	-0.111
						0 848	0 257	0.106	0 191	0 443	0.488	0 364	0 219	0 848	0 568	0.835	0.788	0.488	0 716
						13	13	13	13	13	13	13	13	13	12	13	13	13	13
ID.1							0.124	0.307	0.225	.612(*)	.662(*)	0.46	0 287	-0.27	-0.26	0.076	-0.229	0.361	707(**)
							0.687	0.308	0.46	0.026	0.014	0.114	0 342	0 373	0 414	0.805	0.453	0.225	0.007
							13	13	13	13	13	13	13	13	12	13	13	13	13
ID.2								.721(**)	615(*)	0 425	0	630(*)	669(*)	-0 197	-0 345	0 54	0.388	0.196	0.05
								0.005	0.025	0.147	1	0.021	0.012	0.52	0.272	0.057	0.19	0.52	0.87
								13	13	13	13	13	13	13	12	13	13	13	13
MA.1									0.489	0.197	-0.156	.662(*)	0 474	0 004	-0 216	.620(*)	0.208	0.341	-0.041
									0.09	0.52	0.61	0.014	0.102	0.99	0.5	0.024	0.495	0.254	0.894
									13	13	13	13	13	13	12	13	13	13	13
MA.2										0.179	-0.147	.591(*)	.892(**)	-0.225	-0.43	0.363	0.38	0.177	-0.192
										0.56	0.631	0.033	0	0.46	0 163	0.222	0.201	0.562	0.529
										13	13	13	13	13	12	13	13	13	13
MB.1											.563(*)	0.292	0.325	-0.301	-0.32	-0.042	-0.026	0.046	.620(*)
											0.045	0 333	0 278	0 318	0 311	0.892	0.934	0.875	0 024
											13	13	13	13	12	13	13	13	13
MB.2												0.341	0	-0 552	-0 423	-0.175	-0.339	0.249	850(**)
												0.254	1	0 051	0.17	0.567	0.257	0.412	0
												13	13	13	12	13	13	13	13

	IA.2	IB.1	IB.2	IC.1	IC.2	ID.1	ID.2	MA.1	MA.2	MB.1	MB.2	MC.1	MC.2	MD.1	MD.2	ME.1	ME.2	MF.1	MF.2
MC.1													0.443	-0.517	-.797(**)	0.531	0.153	0.401	0.307
													0 129	0 071	0 002	0.062	0.617	0.174	0 308
													13	13	12	13	13	13	13
MC.2														-0 096	-0.27	0.304	0.367	0.215	-0.123
														0 756	0 397	0.313	0.218	0.48	0.69
														13	12	13	13	13	13
MD.1															673(*)	0.272	0.397	-0.196	-0 519
															0.016	0.368	0.179	0.52	0.069
															12	13	13	13	13
MD.2																-0.054	0.118	-0.051	-0.386
																0.867	0.715	0.874	0.215
																12	12	12	12
ME.1																	.761(**)	0.498	-0.115
																	0.003	0.083	0.709
																	13	13	13
ME.2																		0 243	-0.404
																		0.423	0 171
																		13	13
MF.1																			0.325
																			0.278
																			13
MF.2																			
MG.1																			
MG.2																			
MH.1																			
MH.2																			

	IA.2	IB.1	IB.2	IC.1	IC.2	ID.1	ID.2	MA.1	MA.2	MB.1	MB.2	MC.1	MC.2	MD.1	MD.2	ME.1	ME.2	MF.1	MF.2
RA.1																			
RA.2																			
RB.1																			
RB.2																			
SA.1																			
SA.2																			
SB.1																			

	MG.1	MG.2	MH.1	MH.2	RA.1	RA.2	RB.1	RB.2	SA.1	SA.2	SB.1	SB.2
	0.452	0.487	-0.108	-.597(*)	.628(*)	0.169	0.54	0.498	.586(*)	.664(*)	0.413	.720(**)
IA.1	0.121	0.092	0 726	0 041	0 022	0 58	0.057	0.083	0 035	0 018	0 161	0.005
	13	13	13	12	13	13	13	13	13	12	13	13
	0.5	0.501	-0.09	-.634(*)	559(*)	0.243	0.549	0.347	711(**)	626(*)	0.457	.646(*)
IA.2	0.082	0.081	0.771	0.027	0 047	0.424	0.052	0.245	0.006	0.03	0.117	0.017
	13	13	13	12	13	13	13	13	13	12	13	13
	0.366	0.366	0 066	-0 374	708(**)	0 299	0.508	0.165	762(**)	0 487	0 334	.605(*)
IB.1	0.219	0.218	0.831	0.231	0.007	0.32	0.076	0.59	0.002	0.108	0.264	0.028
	13	13	13	12	13	13	13	13	13	12	13	13
	0.308	0.267	-0.151	-0.397	.737(**)	0.27	.743(**)	0.484	.644(*)	0 552	.563(*)	.787(**)
IB.2	0.306	0.377	0.621	0.201	0.004	0.372	0.004	0.094	0.018	0.063	0.045	-0.001
	13	13	13	12	13	13	13	13	13	12	13	13
	0.378	0.022	0.114	-.597(*)	0.263	-0.34	0.149	0.083	0.193	0.306	0.378	0.397
IC.1	0.202	0.942	0.712	0.041	0 386	0.255	0.627	0.787	0.527	0 334	0.203	0 18
	13	13	13	12	13	13	13	13	13	12	13	13
	0.285	0.437	0.183	-0.162	.676(*)	0.4	0.271	0.179	0.319	0.444	-0.03	0.237
IC.2	0.345	0.136	0 549	0 615	0 011	0 175	0.371	0.558	0 288	0 148	0 922	0.436
	13	13	13	12	13	13	13	13	13	12	13	13
	0.545	0.506	-0.191	-0.235	0 192	0.245	0.255	0.114	0.095	-0 152	-0.18	-0 02
ID.1	0.054	0.078	0.532	0.483	0.53	0.419	0.401	0.712	0.756	0 638	0.555	0.949
	13	13	13	12	13	13	13	13	13	12	13	13
	.595(*)	.587(*)	-0 127	-0.24	0.25	0 384	0.462	808(**)	0 163	0 559	0 289	0 43
ID.2	0.032	0.035	0.678	0.452	0.41	0.222	0.112	0.001	0.596	0.059	0.339	0.143
	13	13	13	12	13	13	13	13	13	12	13	13
	.693(**)	0.478	0.045	-0.174	0 538	.567(*)	.633(*)	.682(*)	0.106	0 374	0.134	0.179
MA.1	0.009	0.098	0.884	0.589	0.058	0.043	0.02	0.01	0.73	0.232	0.662	0.559
	13	13	13	12	13	13	13	13	13	12	13	13
	0.346	.733(**)	-0.34	-0.254	.595(*)	.776(**)	.627(*)	.659(*)	.633(*)	.657(*)	0.319	.626(*)
MA.2	0.248	0.004	0.256	0.425	0 032	0.002	0.022	0.014	0.02	0.02	0.288	0.022
	13	13	13	12	13	13	13	13	13	12	13	13
	0.424	0.526	-0.163	-0.292	0.254	-0.046	0.064	0.233	0.026	0.177	-0.077	0.32
MB.1	0.149	0.065	0 596	0 358	0 402	0 88	0.834	0.444	0 932	0 582	0 803	0.287
	13	13	13	12	13	13	13	13	13	12	13	13
	0.304	0.123	-0.289	-0.516	-0 263	-0.357	0	-0.131	0.101	-0 253	0	0
MB.2	0.313	0.69	0.339	0.086	0 385	0.231	1	0.669	0.742	0 428	1	1
	13	13	13	12	13	13	13	13	13	12	13	13

	MG.1	MG.2	MH.1	MH.2	RA.1	RA.2	RB.1	RB.2	SA.1	SA.2	SB.1	SB.2
MC.1	0.547	0.463	-0.295	-.739(**)	0.397	0.316	.742(**)	.593(*)	.598(*)	0.414	0.478	0.504
	0.053	0.111	0 327	0 006	0.18	0 293	0.004	0.032	0 031	0 181	0 099	0.079
	13	13	13	12	13	13	13	13	13	12	13	13
MC.2	0.526	.650(**)	-0.25	-0.139	0 455	.773(**)	0.407	.568(*)	0.439	631(*)	0.077	0 42
	0.065	0	0.41	0.666	0 118	0.002	0.168	0.043	0.134	0 028	0.803	0.153
	13	13	13	12	13	13	13	13	13	12	13	13
MD.1	0.108	0.062	764(**)	758(**)	-0 076	0 227	-.617(*)	-0.374	-0 364	0 016	-0 523	-0.494
	0.725	0.839	0.002	0.004	0.805	0.455	0.025	0.208	0.222	0.961	0.066	0.086
	13	13	13	12	13	13	13	13	13	12	13	13
MD.2	-0.198	-0.268	0.508	928(**)	-0 297	-0.051	-.586(*)	-0.456	-.581(*)	-0 332	-.656(*)	-.611(*)
	0.536	0.399	0.092	0	0.348	0.874	0.045	0.137	0.048	0.318	0.02	0.035
	12	12	12	12	12	12	12	12	12	11	12	12
ME.1	.605(*)	0.372	0.354	0.122	0.17	0.308	0.209	0.393	0.164	0.272	-0.09	0.016
	0.028	0.21	0.235	0.706	0 576	0.216	0.493	0.185	0.592	0 392	0.771	0.959
	13	13	13	12	13	13	13	13	13	12	13	13
ME.2	0.211	0.393	0.245	0.266	0.137	0.342	-0.07	0.205	0.244	0.477	-0.17	0.144
	0 49	0.184	0 421	0 404	0 655	0 253	0.819	0.502	0 422	0 117	0 579	0.638
	13	13	13	12	13	13	13	13	13	12	13	13
MF.1	0.379	0.317	-0.072	-0.058	0.04	0.287	0.225	0.158	0.102	-0 197	-0.473	-0.334
	0.201	0.291	0.816	0.858	0 896	0.342	0.461	0.606	0.741	0.54	0.102	0.264
	13	13	13	12	13	13	13	13	13	12	13	13
MF.2	0.308	0.088	-0 245	-0 368	-0 264	-0 382	0.026	0.051	-0 182	-0 506	-0 122	-0.121
	0.306	0.774	0.419	0.24	0.384	0.225	0.934	0.867	0.551	0.093	0.692	0.695
	13	13	13	12	13	13	13	13	13	12	13	13
MG.1		.593(*)	0.307	-0.066	0 246	0.369	0.195	0.368	0.021	0 192	-0.143	-0.007
		0.033	0.308	0.839	0.418	0.215	0.524	0.217	0.945	0.55	0.642	0.982
		13	13	12	13	13	13	13	13	12	13	13
MG.2			-0.071	-0.129	0.437	.748(**)	0.19	0.364	0.396	0.562	-0.176	0.231
			0.818	0.688	0 136	0.003	0.533	0.222	0.181	0 057	0.566	0.448
			13	12	13	13	13	13	13	12	13	13
MH.1				.593(*)	-0.051	-0.103	-.587(*)	-0.454	-0.263	-0.037	-0.409	-0.392
				0 042	0.87	0 738	0.035	0.119	0 385	0.91	0 165	0.185
				12	13	13	13	13	13	12	13	13
MH.2					-0 259	0.058	-.591(*)	-0.311	-.579(*)	-0 376	-.660(*)	-0.544
					0 415	0.858	0.043	0.326	0.049	0 255	0.02	0.067
					12	12	12	12	12	11	12	12

	MG.1	MG.2	MH.1	MH.2	RA.1	RA.2	RB.1	RB.2	SA.1	SA.2	SB.1	SB.2
RA.1						.556(*)	.579(*)	0.324	0.537	0.566	0.19	.577(*)
						0 048	0.038	0.281	0 058	0 055	0 534	0.039
						13	13	13	13	12	13	13
RA.2							0.4	0.411	0.355	0.47	-0.117	0.112
							0.176	0.164	0.234	0 123	0.704	0.716
							13	13	13	12	13	13
RB.1								743(**)	0 463	0 309	592(*)	.589(*)
								0.004	0.111	0.329	0.033	0.034
								13	13	12	13	13
RB.2									0.086	0 281	0.424	0.482
									0.78	0.375	0.149	0.096
									13	12	13	13
SA.1										.751(**)	0.47	.674(*)
										0 005	0.105	0.012
										12	13	13
SA.2											0.451	.698(*)
											0 141	0.012
											12	12
SB.1												.770(**)
												0.002
												13

3: Raw response data

A2	A3	A4	A5	A6	G1	G2	G3	ID1	ID2	IA.1	IA.2	
3	4	4	3	3	5.5	1	4	1	4.5	4	5	4
13	5	5	5	4	5.5	1	4	1	5	5	5	5
19	4	4	5	5	5.5	1	5	2	5	5	5	5
30	3	4	4	3	5.5	1	5	4		5		5
37	5	5	3	3	5.5	1	4	1		3.5		4
33	3	4	5	4	3.5	1	5	4		5		5
35	5	5	4	5	5.5	1	5	2		4.5		4
39	5	5	4	5	3.5	1	5	2		5		5
24	3	4	4	4	5	2	5	1	3.5	4	4	4
4	5	5	5	3	5.5	2	4	2	5	5	5	5
6	5	5	5	5	5.5	2	4	1	5	5	5	5
29	2	2	2	4	5	2	5	1		4		4
11	2	2	2	5	1.5	3	5	3	3.5	3.5	4	4
18	4	5	2	5	2	3	5	3	4.5	4.5	4	4
31	4	4	4	3	5.5	3	4	1		4		4
36	4	4	3	4	2.5	3	5	5		4		4
32	4	5	4	5	5	3	4	3		3.5		4
17	5	5	2	5	5	4	3	1	2.5	2.5	3	3
10	3	3	3	3	2	4	4	2	4	3.5	4	3
8	3	4	3	4	3	4	5	2	4.5	4	4	4
9	4	4	3	5	2	4	3		4	4	4	4
34	4	5	4	5	3	4	5	2		4.5		4
38						4	5	5		4		4
22	5	5	1	5	3	5	5	2	3.5	3.5	4	3
1									3.5		3	
7									2.5		3	
25									3.5		3	
2									4		4	
5									4		4	
12									3.5		4	
14									3		4	
16									4.5		4	
20									4		4	
21									4.5		4	
23									4.5		4	
26									4		4	
15									5		5	
27									5		5	
28									4.5		5	

IB.1	IB.2	IC1	IC2	IC.1	IC.2	ID.1	ID.2	MI1	MI2	ME1	ME2	MA.1	
	4	4	2.5	4	3	4	2	4	2	4	2.75	3	2
	5	5	2	4.5	2	4	2	5	4 25	4 5	2 25	2 5	5
	5	5	4	4.5	3	4	5	5	4.25	4	2.25	3	5
		5		4		4		4		3 5		2.5	
		3		2		2		2		2 5		2.5	
		5		1		1		1		4		1.5	
		5		3		2		4		4.25		3.25	
		5		4		4		4		4		2.75	
	3	4	2	2.5	2	1	2	4	1.75	3.75	1.75	3.5	2
	5	5	2.5	4	3	3	2	5	5	4.75	2 25	2.25	5
	5	5	3.5	4	2	3	5	5	5	5	3	5	5
		4		3		2		4		4		3.25	
	3	3	3	2.5	3	2	3	3	3	2	2 75	3.75	3
	5	5	2.5	2	2	2	3	2	2	2.75	1.5	3	2
		4		3.5		3		4		3		2.5	
		4		3		3		3		3.25		2.5	
		3		2.5		2		3		3.25		2.75	
	2	2	2	2	1	1	3	3	2 25	4	2 25	3 5	2
	4	4	2	3.5	2	3	2	4	2.5	3.5	2.5	4	2
	5	4	2.5	3	2	4	3	2	2	3.75	2 75	3.5	2
	4	4	2.5	3.5	2	3	3	4	4.25	4.25	3 75	4	5
		5		3		3		3		4		3	
		4		4		4		4		3		3 5	
	3	4	2	4	2	4	2	4	2.75	3.25	2	3.25	5
	4		2.5		2		3		4		2.5		5
	2		2.5		2		3		2		2		2
	4		1.5		1		2		3.25		2.75		5
	4		2		1		3		3.5		3 25		5
	4		3		3		3		2.25		2.25		3
	3		1.5		1		2		2.25		2		2
	2		2.5		3		2		2		2		2
	5		1.5		1		2		2.25		1.75		2
	4		1		1		1		2.5		2		3
	5		3		3		3		2		2.5		2
	5		1		1		1		2		2.5		2
	4		3		3		3		5		2 25		5
	5		2		2		2		3.25		2.5		2
	5		3		3		3		4		2 75		5
	4		2		2		2		2		1.5		2

MA.2	MB.1	MB.2	MC.1	MC.2	MD.1	MD.2	ME.1	ME.2	MF.1	MF.2	MG.1	MG.2	
	4	5	4	2	4	2	2	2	4	2	4	2	4
	5	2	3	5	4	2	2	5	5	3	3	2	4
	5	5	5	5	5	1	1	2	1	2	5	5	5
	4		4		4		2		3		3		3
	3		3		2		2		3		4		2
	5		2		1		1		5		2		5
	4		1		4		4		5		4		4
	4		4		4		3		4		2		4
	4	3	4	2	4	2	4	2	4	1	3	1	3
	5	2	4	5	5	2	1	5	5	3	3	5	4
	5	5	5	5	5	1		5	5	5	5	5	5
	4		1		3		4		5		4		4
	2	3	5	3	2	2	3	3	2	3	5	3	2
	4	2	4	3	3	1	2	2	2	2	4	1	2
	3		3		3		2		3		3		3
	4		3		4		3		3		2		2
	4		2		3		3		2		4		4
	4	2	4	2	4	3	3	3	4	3	4	2	4
	4	2	4	2	4	2	4	3	3	3	4	3	3
	4	2	4	2	4	3	4	2	3	3	3	2	4
	4	3	3	2	4	5	5	5	5	2	3	5	4
	4		4		4		2		4		4		4
	3		3		3		4		3		3		3
	4	2	3	2	4	2	4	2	2	3	3	2	3
		2		5		2		3		3		3	
		2		2		2		2		2		2	
		3		3		3		2		2		3	
		5		5		3		2		2		2	
		2		2		2		2		2		2	
		3		2		1		2		2		3	
		2		2		2		2		2		2	
		2		2		1		2		3		3	
		2		2		2		3		2		2	
		2		2		3		2		2		2	
		3		2		2		2		3		2	
		5		5		1		5		2			
		2		5		2		3		3		3	
		5		5		2		3		3		3	
		2		2		1		2		2		2	

MH.1	MH.2	MX1	MX2	RA.1	RA.2	RB.1	RB.2	S	231 Grade	233 Grade	SB.1	SB.2	
	2	2	2	3	5	4	2	4	4.25	4	4 5	3	5
	2	2	2.25	2.75	5	5	5	5	5.5	5.5	5.5	5	5
	1	1	2.25	3	5	5	5	5	4.5	4.5	4 5	5	5
		1		2.25		5		5	4.5		4.5		4
		1		2.25		2		2	5		5		4
		1		2.25		5		3	2		2		5
		4		4		4		4	3.5		3 5		5
		2		2 75		5		5	4		4		5
	1	3	1.25	3.25	3	4	3	4	4.5	4	5	5	5
	2	1	3	2.25	5	5	5	5	5.5	5.5	5 5	5	5
	1		3	5	5	5	5	5	4	4.5	3 5	2	4
		4		4		4		4	4.5		4.5		4
	3	2	2.75	3	2	3	2	3	2	2.5	1 5	3	2
	1	2	1.25	2.5	5	4	5	4	3.25	4.5	2	5	5
		2		2.5		4		2	4		4		4
		2		2.25		5		5	3		3		4
		2		3 25		4		3	5		5		2
	1	3	2.25	3.5	2	5	2	4	2.5	3	2	2	2
	3	4	2.75	3 75	2	4	2	4	2 75	3	2 5	3	3
	3	3	2.75	3.5	5	5	2	2	5	5.5	4.5	2	3
	5	5	4.25	4 25	5	5	2	4	2.5	2 5		2	3
		2		3		5		5	4		4		4
		4		3.5		3		3	3		3		3
	1	3	2	3.25	5	5	5	5	2.75	2.5	3	3	3
	3		2.75		1		2		3	3		3	
	2		2		2		2		4.5	4.5		5	
	3		2.75		3		2		4	4		5	
	3		2.5		5		2		4.5	4.5		2	
	3		2.25		2		2		2.5	2.5		3	
	2		2		3		2		2	2		1	
	2		2		5		5		3.5	3.5		3	
	1		2		5		5		5	5		2	
			2										
	3		2.5		5		5		4.5	4.5		2	
	2		2.25		5		5		3	3		5	
	1		1.333333		3		2		4	4		5	
	3		2.75		5		5		4.5	4.5		2	
	1		2.25		5		5		5	5		5	
	1		1,5		2							3	