MEASURING THE IMPACT OF APP INVENTOR FOR ANDROID AND STUDIO-BASED LEARNING IN AN INTRODUCTORY COMPUTER SCIENCE COURSE FOR NON-MAJORS

A DISSERTATION

SUBMITTED TO THE GRADUATE SCHOOL

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE

DOCTOR OF EDUCATION

BY

KHULOUD NASSER AHMAD

DISSERTATION ADVISOR: DR. PAUL V. GESTWICKI

BALL STATE UNIVERSITY

MUNCIE, INDIANA

JULY 2012

ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 - 1346

ABSTRACT

A reexamination of the traditional instruction of introductory computer science (CS) courses is becoming a necessity. Introductory CS courses tend to have high attrition rates and low success rates. In many universities, the CS department suffered from low enrollment for several years compared to other majors. Multiple studies have linked these phenomena with low student motivation, specifically with respect to attitudes towards overtly mathematics-oriented assignments and lecture-oriented pedagogy. Students' criticisms were directed at the major for its lack of creativity, relevance, and interest. This study implemented an experimental introductory CS course for non-CS majors focusing on two pedagogic factors: 1) the use of a visual blocks programming language known as App Inventor for Android (AIA) and 2) the adoption of SBL as the main teaching methodology. Participants included 30 undergraduates enrolled in two introductory CS courses; the experimental course (CS116) and a traditional lecture oriented CS course. The Motivated Strategies for Learning Questionnaire (MSLQ) was implemented in both courses at several stages. Statistically significant differences were found in the Control of Learning Beliefs, Help Seeking, and Intrinsic Motivation scales, were CS116's participants scored higher rates. In CS116, entry and exit interviews were conducted as well as a mind maps analysis. Their results showed a positive response to the pedagogic factors, positive attitudes towards CS, and an improvement in the understanding of CS. The majority of participants did very well and showed creativity with not one student failing the course. They found the experimental course to cultivate collaboration, creativity, and motivation to learn. The experimental approach was found have a positive effect on students' motivation, achievement, and attitude towards CS.

ACKNOWLEDGMENTS

My sincere thanks are due to Google and the AIA development team for their contributions and support throughout the implementation of this study; special thanks to Google for providing the G1phones used in this study. My gratitude is extended to Dr. Hal Abelson for his effort and continuous support.

I am especially indebted to Dr. Paul Gestwicki for his guidance in formulating the foundation of the study and for his continuous guidance, support, and good counsel throughout the dissertation process. I am indebted to many whose exemplary teaching, guidance, and support have been an inspiration for me and my study in the United States.

Special thanks to my parent for guiding and supporting me throughout my education. They have always encouraged me to be myself and to pursue my dreams. The independence they have instilled in me drove my pursuit for higher education.

Last, but definitely not least, I would like to thank my other half, my beloved husband Hamid, who was always by my side and who put up with all my irritability that resulted from the stress of writing this dissertation. I would like to thank him for his support, both emotionally and financially, throughout this process. Without him, none of this would be possible. Hamid, now you have my admission in writing :)

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION

A reexamination of the traditional instruction in introductory computer science (CS) courses is becoming a necessity. In many universities, the CS department has suffered from low enrollment for several years [81], particularly since the *dot-com bust* in 2000-2001 [39,86]. Enrollment in CS was at its highest rate in 2001 after which it started experiencing a steady and rapid decline [86]. From 2000 to 2007, enrollment in CS has declined by 49% and interest in the major among incoming undergraduates has declined by even more than that (see Figure 1-1). There was a slight increase of 8% in enrollment in 2008 and then a 5.5% in 2009 [91,92]; these increases, however, are not enough to bring back the major from its steep decline. In 2010, enrollment in CS stayed about the same as the year before [93]. Cooper and Cunningham [19] argue, as well, that there are "some promising results here and there, but an overall pattern of change and improvement is not yet evident" (p. 5).

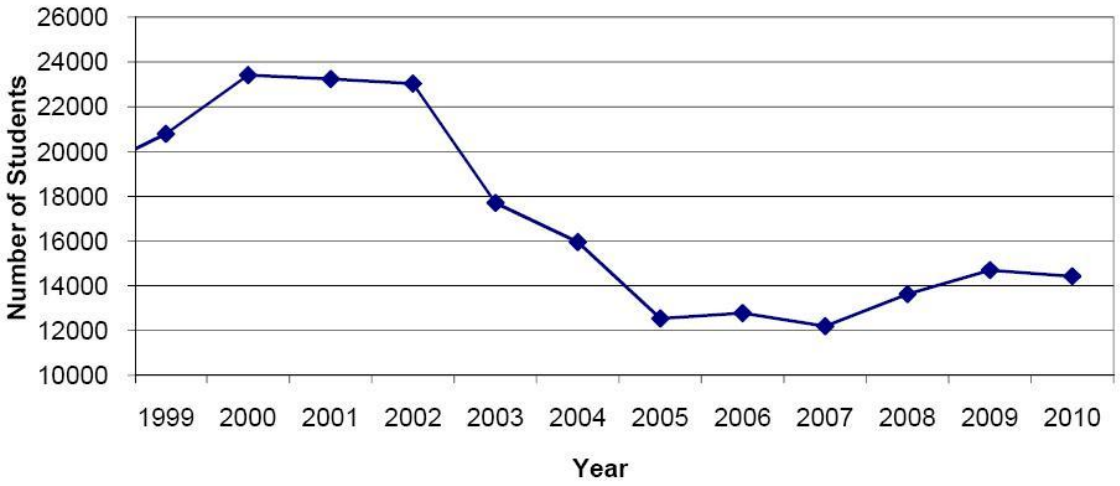


Figure 1-1: ***Trends of Enrollment in CS Major*** **[93].**

Compared to other majors, the CS major has very high attrition rates [81]. Cooper and Cunningham [19] found that "there is little evidence that our students complete courses or stay in their degree programs" (p. 5). Cohoon and Chen [18] as well as Sloan and Troy [81] stated that this rate in the U.S. among freshmen and sophomores was 19%, and at some schools it was as high as 66%. In addition to the high attrition rates, introductory CS courses tend to have low success rates [18,27,81]. Multiple studies have linked these phenomena with low student motivation, specifically with respect to attitudes towards overtly mathematics-oriented assignments and lecture-oriented pedagogy [27,73]. Students' criticisms are directed at the major for its lack of creativity, relevance, and interest [33,73]. However, according to the Bureau of Labor Statistics (BLS) there is still an increasing demand for workers with technical computer skills and computer related services [13,37]. BLS anticipates the demand for computer programmers to increase by 12% between 2010 and 2020 [12]. In fact, an increase of

22% is anticipated for computer science related occupations as well. BLS ,furthermore, found computer systems design and related services to be one of the fastest growing industries where its demand is projected to increase by approximately 31% over the next eight years [37]. BLS expects that Computer Programmers' "job prospects should be best for those with a bachelor's degree and experience with a variety of programming languages and tools" (p. 3) [11]. To improve such prospects, BLS suggest being current with the latest and newest programming tools [12]. Therefore to provide CS graduates with better job prospects, they need to be versatile programmers, that is be able to adapt to any programming language.

The problem seems to be with students who have just embarked on the major. Forte and Guzdial [27] found that "traditional introductions to computer science are more likely to frustrate students than attract them to the field" (p. 2). Studies showed that the introductory CS course failed to reach the wide range of students taking the course, it had a decreased enrollment of female students, and students conceded the course to be "overly-technical," not applicable in the real world, and lacking opportunities for creativity [33]. According to Freudenthal, Roy, Ogrey, Magoc, and Siegel [28], "programming techniques in early [introductory] courses should be chosen to minimize cognitive load while maximizing pedagogical value" (p. 37). Forte and Guzdial [27] stated that the reason behind students failing and dropping out of CS was not perceiving it as interesting or useful. Female students, "*in contrast to men,* are mostly interested in real applications of computing, and not simply computing for its own sake" (p. 271) [81].

1.1 Creativity

The industry noted "a lack of creativity in school graduates" (p. 149) [73]. Research on creativity in CS education is very rare [48]. One study found the traditional computer science education approach focused on problem solving and lacked creative activities [73]. Other research papers found that CS students struggle with tasks that involve creativity [4, 30]. Gottel and Schild [30] found that students are "not well-trained in creative thinking" (p. 98). They observed "discomfort amongst students when asking them to realize their own ideas" (p. 98).

Creativity is hard to define and thus hard to measure. Many have attempted to classify creativity in order to find tangible grounds on which to measure. Romeike [73] defined creativity as "something that is fundamentally novel to the individual" (p. 149). He bases this definition on Margaret Boden's *P-creativity*: the psychological sense of creativity [8]. Boden describes another sense of creativity known as *H-creativity*: the historical sense of creativity. While the former involves ideas that are new to the creator, *H-creativity* involves ideas that are new to the history of creation. In a classroom environment *P-creativity* is more relevant. Students are still learning the concepts and basics of their discipline and therefore cannot be expected to have ideas that are historically novel, ideas that supersede those who have more experience and knowledge in their field. Another definition that relates creativity to educational settings is Fink's [26]. He defines creative thinking as the process of "find[ing] new ways of answering questions, . . . , or to devise new solutions to old problems" (p. 42) with "new" signifying its novelty to the student and his immediate environment. Romeike [74] goes

further to define three dimensions of creativity in CS: PERSON, SUBJECT, and ENVIRONMENT (the capitalization of the terms comes from [74]). The PERSON-dimension is the intrinsic motivation and its relationship to creative performance. The SUBJECT-dimension is the creative processes involved in the software development process. The ENVIRONMENT-dimension is the impact of the tools used and their support for creativity. All three dimensions have a significant impact on CS education and creativity within it.

A positive relationship is found between creativity and motivation. Romeike [73] found that creative activities increase students' motivation thus enhancing their understanding of CS concepts. Such creative activities cultivate creative characteristics in the students, such as fluency, flexibility, and creative problem solving. Similarly, Knobelsdorf and Romeike [48] found that high creativity is often associated with a desire to learn, explore, and understand. Lewandowski, Johnson, and Goldweber [52] found that not only does creativity increase motivation but it also increases interest in CS and as a consequence improves retention. Therefore, a new approach that incorporates activities that foster creative thinking needs to be devised.

1.2 Multimedia and Mobile Technology

New and innovative approaches are being implemented and studied to combat these problems in introductory CS courses. One implementation is the media computation approach that entails the introduction of CS concepts through the use of multimedia. Computation and multimedia complement each other, for "digital media are computationally created and manipulated" (p. 2) [27]. In a media computation class,

students learn the fundamentals of programming through manipulating media, such as pictures, sounds, movies, and web pages. In using their own media, students get to "use computation in a personally expressive way" (p. 3) [27] that is at the same time engaging; keeping students engaged improves retention rates.

Media computation creates a "creative context" that has a positive effect on interest in computer science [27]. The media computation approach provides relevancy to the students. It is part of everything they do in their daily lives. Multimedia is what they spend most of their time on. Making multimedia part of their class experience makes common sense. However, CS courses are perceived to be "overly-technical" [33], therefore simply introducing media in the traditional CS courses may not be sufficient. The way the material is introduced needs to be reexamined as well; a reexamination that provides simpler methods of getting the basics across without spending much-needed time on marginal details. Researchers argue that the current social context of the media plays as big a role as its capabilities in its use in learning computation [27,49]. How media is being used in social situations must be taken into consideration when contemplating how it will be incorporated into a computation learning environment. These factors make implementing this approach a complex matter and they need to be considered in order to make the implementation a successful one.

Mobile technology is one of the places where multimedia is often utilized. Mobile application development is a growth area of computer science, and mobile devices are useful as a motivational tool to stimulate interest [50]. Tew, Fowler, and Guzdial [83] emphasized the importance of understanding the students and what their interests are. These considerations make the learning process more relevant to the students. They

advocated using media in a context students are familiar with and adhere to the constructivist approach in that people learn when they construct artifacts that are meaningful to them [27]. Context, on its own, provides students with motivation and meaning as well [20]. John Dewey, a renowned psychologist and education reformer, argued this decades ago. Dewey argued that students find a problem to be meaningful when it relates to their personal interests and experiences [22]. Cooper and Cunningham [19] argued that "building our courses within a context may increase learning, increase motivation, and ultimately increase the number of students who are attracted to computer science and seek to remain in the field" (p. 6).

Today, mobile technology is an integral part of students' lives, therefore, they relate to mobile applications easier and their use is meaningful to them [50, 72]. Kurkovsky [50] found that developing mobile applications "offer[s] instant gratification" (p. 47) to the students. They "can download them to their mobile phones almost immediately and show them off to their friends" (p. 47) [50]. However, mobile applications are not easy for students to write and the tools available are too complicated. Until now, students required many semesters of education before they were able to write mobile applications.

1.3 Programming Language

The practice of using a professional-grade development language in introductory CS courses started when languages were easy for beginners [21]. Today, Java and C++, two professional-grade development languages, are standard languages in the industry and are the standard languages used in introductory CS courses. However, these

languages were created for developing large-scale software and not intended for instructional settings [69]. In using those, students struggle, get distracted, and frustrated. For instance, Freudenthal et al. [28] used the object oriented Java AWT toolbox as the programming interface of an introductory CS course. They found that "the *conceptual* content embedded within . . . introductory programming lessons were often overwhelmed by the mainly *critical* task of managing the access and manipulation abstractions for pixels in Java" (p. 38). According to Denning, "students are being overwhelmed by the complexities of languages that many experts find challenging . . . . Many students have turned to cheating and plagiarism as ways to pass these courses, and 35%-50% drop out prematurely" (p. 20) [21]. Meyer and Masterson [62] found that "students in 'CS 0' courses and even early programming courses find algorithm design difficult and seldom much fun" (p. 184).

In an effort to combat these problems, a new trend was starting to emerge in introductory CS courses; Languages such as Python, Alice, and Scratch are not a standard in the industry but are being used as languages of introductory CS courses - for the purposes and within the context of this study such languages will be referred to as *pedagogic language*s. Such languages are easier to understand for the novice and are more suitable as the medium for introducing and implementing CS concepts than professional-grade programming languages, such as Java and C++. In other developed CS introductory courses Java was still the language of choice but it was introduced through a lightweight development environment known as DrJava [35]. The feature in DrJava that is beneficial for novice programmers is its *instructions pane* (see Figure 1-2). This pane acts like an interpreter for Java which allows teachers to introduce basic

programming constructs without having to introduce complex Java concepts such as classes.

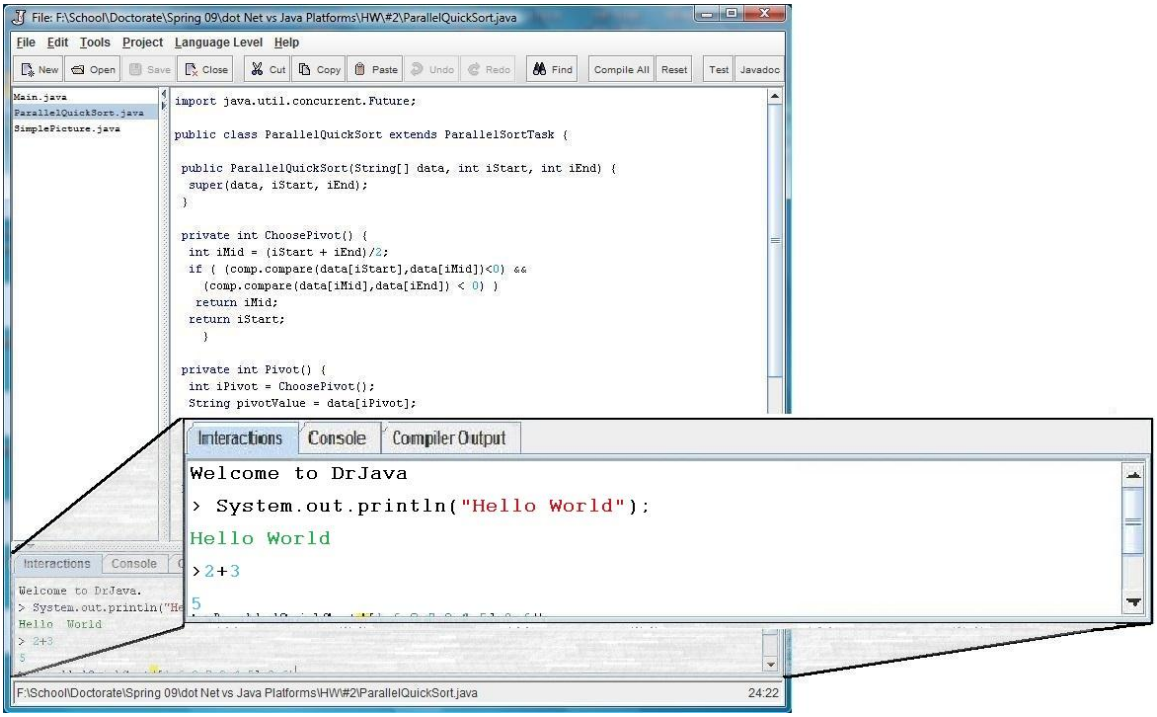


Figure 1-2: ***The Instructions Pane Feature in DrJava.***

Some courses were developed that use textual pedagogic languages such as Python; other courses used languages that depend on a visual–or graphical–environment rather than textual syntax, for example Alice and Scratch (see Figure 1-3). These languages are also known as visual blocks programming languages; since a program is built through snapping together building blocks. Surprisingly, novice programmers often find difficulty in understanding the simple concept of consistent indentation [62]. Where in visual blocks programming languages, blocks fit together a certain way using drag and

drop actions, much like a puzzle, creating the sequence of the program eliminating the need for indentations all together; thus eliminating the struggle students have with language syntax and the trivial syntactical errors. Overall, Meyers and Masterson [62] found that the "visual metaphor of containment and subordination of control" (p. 184) in visual blocks programming languages aid in understanding algorithmic design principles. It is important to note that even though visual programming significantly helps with accuracy and reducing bugs, it is not the debugging panacea [87]. The measure of its aid is highly dependent on the problem being solved, the user solving it, and the type of bug. However, the debugging speed over time was found to be slightly faster using visual programming languages.

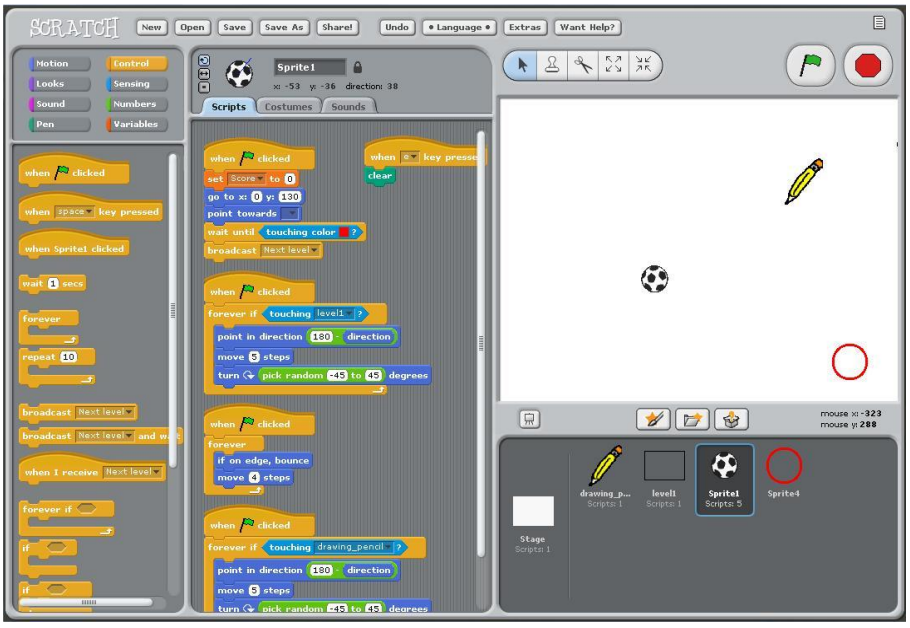


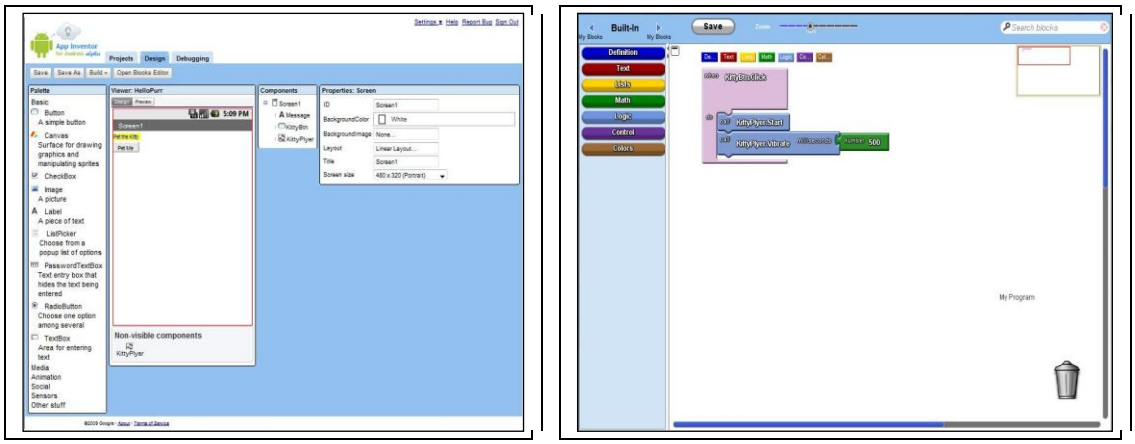
Figure 1-3: ***Scratch Development Environment.***

1.3.1 App Inventor for Android

According to Meyer and Masterson [62], there are three types of visual programming languages. First, is the imperative language that has a graphical user interface (GUI) based design environment and focuses on building GUI applications, Visual C++ for example. Second, is the true visual programming language that emphasizes a specific problem area such as modeling and simulation, LabView is one example. Third, is the visual blocks programming language which is considered "the true general-purpose visual programming language" (p. 183), in that category is Alice and Scratch. However, Alice and Scratch are environments for developing applications geared toward animation [6, 80]. Scratch, as well, is directed at a younger audience, which is elementary through high school [80]. Therefore, there was a need for an environment that would be suitable for teaching programming concepts in entry/introductory level courses for undergraduates and not specific to a certain theme.

The pedagogic language chosen for this study was a recently developed visual blocks programming language known as the App Inventor for Android (AIA) (see Figure 1-4). AIA is a relatively new technology from Google, Inc, geared towards introductory CS education [5]. It is free and publicly available. AIA is the first visual blocks programming language created for educational purposes that develop applications exclusively for mobile devices that run on the Android operating system. Its purpose was to allow students with no prior programming experience to easily develop mobile applications for mobile devices, specifically the G1 phone at the time of this study (see Figure 1-5). AIA was chosen over other visual blocks programming languages for this

study due to its mobile application development capability. It allows instructors to leverage students' interest in mobile applications and the ubiquity of mobile devices in order to increase students' motivation, compared to traditional approaches to teaching introductory CS. The students created applications that manipulate media, making use of the media tools that phones typically provide. Unlike its counterparts, that is Alice and Scratch, AIA has the capability of developing all types of applications not just animation. It has not been implemented till this study nor is there any available empirical information that bears upon it.



a. Application Designer b. Blocks Editor

Figure 1-4: ***App Inventor for Android.***

Figure 1-5: *G1 Phone* [www.htc.com].

The AIA development environment is a World Wide Web integrated development environment (IDE). It is accessed through a World Wide Web browser and does not require any installation or setup. At this testing phase, however, access was restricted to authorized users. Additionally, those authorized users needed a Google account to be able to use the development environment. Currently, authorization is no longer required; AIA is now open to the public and all that is required is a Google account.[1] The IDE consists of two parts, the Application Designer and the Blocks Editor (Figure 1-4). In the Application Designer, the application developer would select and organize the components that form the application's graphical user interface (GUI). The Blocks Editor is where the developer selects and organizes blocks in the desired sequence. The visual blocks components available in AIA cover basic CS concepts such as control statements, loops, logic programming, strings, and lists. It also has a library of functions providing capabilities such as string and list manipulations and math functions. The Blocks Editor

[1] For more information go to http://www.appinventor.mit.edu/

uses OpenBlocks[66] as the bases for its visual blocks. OpenBlocks is a Java library for creating visual blocks programming languages distributed by MIT's Scheller Teacher Education Program.[2] After creating the blocks sequence, the compiler then translates it for implementation on Android OS by creating a QR code. This compiler uses the Kawa Language Framework and Kawa's dialect of the Scheme programming language, developed by Per Bothner and distributed as part of the Gnu Operating System. The QR code is a 2D barcode that stores a hyperlink to the compiled code on an online server. The G1 phone would then scan the created QR code using a barcode scanner application, which utilizes the phone's camera, and then provides the hyperlink. Clicking on the link would download the compiled code and install it on the G1 phone. An emulator can be integrated that mimics the phone and provides a possibility for use in teaching cases where mobile devices are not available. It is important to note, however, that the G1phones used in this study are now considered ancient technology. Fortunately, the current AIA IDE creates application that would run on any mobile device that runs on the Android operating system.

1.4 Studio-Based Learning

The traditional method of instruction in CS courses in general is insufficient to prepare students adequately for CS as a profession [42]. In addition to programming skills, the profession is in need of skills in communication, collaboration, and critical thinking [42]. According to Docherty, Sutton, Brereton, and Kaplan [23], CS courses are often taught using methodologies used for mathematics and engineering. Cooper and

[2] For more information go to http://education.mit.edu/drupal/

Cunningham [19] believe that "mathematics education focused on formal theory and techniques with little connection to the many areas where those theories'[sic] or techniques'[sic] come from or where they can be used" (p. 5). Docherty et al. [23] argue, however, that CS has characteristics that make it different from those two disciplines; for example, the criteria for measuring success in a CS course are unclear. They claim that CS as a discipline is closer to those of design sciences, such as architecture. Thus, they argue that it should follow teaching methodologies used in such design disciplines, for example Studio-Based Learning (SBL). Carter and Hundhausen [17] agree with this argument and "believe the SBL model is readily adaptable to the computer science discipline" (p. 106). Hence, another approach that is being implemented in CS courses today, including introductory, is the Studio-Based Learning (SBL) approach.[3]

Derived from the architecture discipline [9], SBL is a pedagogic approach that applies techniques from architecture and art to Science, in this case computer science (see Figure 1-6). SBL is a derivative of problem-based learning (PBL) [60]. Problem-based learning involves stating a problem to be solved, students most likely working in groups, and possibly, at the end, solutions being discussed. Studio-Based Learning is similar to problem-based learning but adds to it a focus "on the artifacts created by the students as the basis for discussion and further work" (p. 273) [60]. The discussion sessions, known as *design crits* (short for critiques), that take place after the process of finding solutions to the problem are key to Studio-Based Learning. The main goal in both of these two learning styles, SBL and PBL, is to mix the theory being studied with actual practice and

[3] For more information see http://studiobasedlearning.org/

implementation. According to Carter and Hundhausen [17], SBL contains four key characteristics. They are the following:

1. Programming assignments given in the classroom are project-based,
2. Through the *design crits*, both formal and informal evaluations of students' work are done periodically,
3. Similarly, students engage in peer evaluation and critique,
4. And *design crits* "should revolve around the artifacts typically created by the discipline" (p. 106) [17].

SBL facilitates the feedback process, which is essential to any learning process, making it more of an immediate feedback.

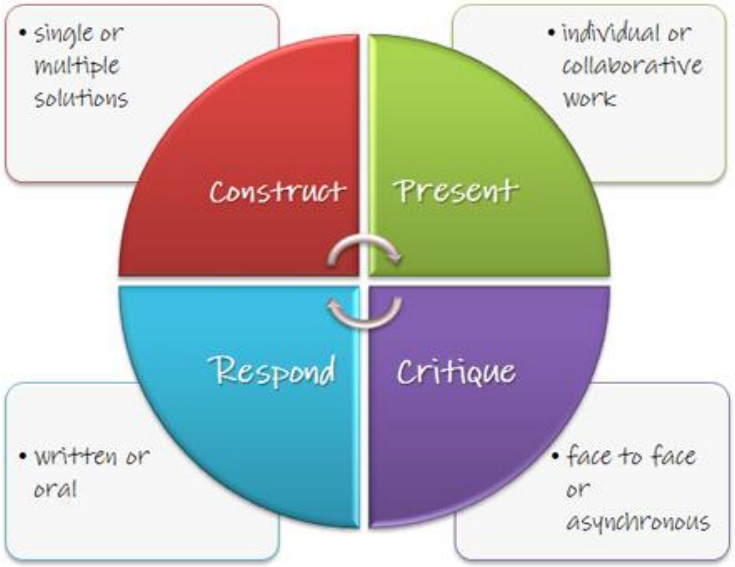


Figure 1-6: ***The Studio-Based Learning Model*** **[79].**

As a form of laboratory instruction, SBL insures active learning [60]. Active learning focuses on keeping students active and involved in the learning process. Active learning, moreover, not only helps the students learn the material being taught, but also helps them learn how to think as well, and to acquire knowledge [75, 60]. It turns learning into an intrinsic goal for the students which, in turn, increases their motivation to learn and transforms them into active learners even outside of the educational institution. Studio-Based Learning (SBL) focuses on creating an active classroom environment where concepts are introduced and better retained; an environment where the students are active, involved, and interacting with the instructor and, more importantly, with other students. Many research papers have shown that students learn more from their peers than they learn from their instructor [60].When compared to traditional lecture-based courses, SBL was found to be more beneficial in terms of grades and scores and in terms of students' attitudes [17].

Bloom's taxonomy is a scheme, involving the cognitive domain, "for classifying [course objectives and] question sets and problems for classroom discussion or examinations" (p. 27) [75]. It classifies them into six levels ranging from simple to most complex [70]. SBL meets almost all levels of Bloom's taxonomy [82]; giving a more comprehensive learning experience. SBL, moreover, incorporates application, analysis, synthesis, and evaluation levels; the higher levels in the taxonomy [82]. Traditional lecture oriented teaching does not go beyond the first two levels in Bloom's taxonomy [70, 75]. Those two levels combined involve only memorizing and understanding, they do not extend the learning experience to include applying and analyzing.

One of the reasons posited for the low interest in CS and the low retention rates is the anti-social image the discipline has [40], the image of a computer scientist as a socially-challenged person sitting in front of the computer all day. Hundhausen et al. [40] claim that this image is due to the association of computing with programming, since many early CS courses tend to focus mainly on programming and learning the syntax of a language. The industry does demand proficient programmers which makes this focus understandable and even necessary. However, even after taking many CS courses that focus on programming, students still have poor programming skills [90]. Woodley and Kamin [90] argue that learning skillful programming is "much like learning to write well: the student needs to receive detailed feedback, rewrite, and receive more feedback" (p. 531). Writing may not be the best analogy here, since it's mostly done in isolation; CS professionals work in design teams, not basements, so collaborative skills are essential. What can be inferred here, however, is that programming requires lots of practice with feedback, similar to writing. This entails that the instructor would have to meticulously evaluate each assignment and write ample feedback and repeat the process again. That process would be exhausting as well as time consuming. Here Studio-Based Learning, with its social setting and its periodical *design crit* sessions, may offer a solution. Lewis et al. [53] argue that "integration of cooperative and collaborative learning might challenge cultural stereotypes and set more accurate expectations of CS professions" (p. 10).

Hundhausen et al. [40] finds SBL to be a natural fit for computing courses at all levels. One of the reasons why most studies implement the SBL approach is "to give students exposure to industry practice" (p. 110) [17]. Recent studies have shown that

SBL improves student motivation and interest in CS, as well as helps students develop their communication skills [40]. Students have generally been positive about learning experiences in studio environments [16], and although longitudinal impact studies of the impact of SBL are still underway, preliminary results are positive.[4] Overall, these implementations showed great promise in addressing the problems in introductory CS courses.

It is worth mentioning, however, that most studies that implemented SBL so far used "SBL to reinforce lecture material" (p. 110) [17]. The reverse is worth investigating; where the main teaching methodology used in the course would be SBL and lectures are used to reinforce the SBL sessions. There is a need for "additional comparative studies" (p. 110), as Carter and Hundhausen [17] believed, where SBL is compared to other methodologies. They believe, moreover, "the interplay between lecture and SBL" (p. 110) should be further studied. This study further investigates this interplay focusing on SBL as the main teaching methodology and using, very few, lectures only as reinforcement to the SBL sessions.

1.5 Purpose of the Study

This study used the above approaches to address some of the problems in introductory CS courses, such as lack of interest, lack of creativity, programming language complexity, and the anti-social learning environment. The pedagogic language used in this experimental study was a visual blocks programming language that creates mobile applications known as the App Inventor for Android (AIA). The main teaching

[4] For more information see http://studiobasedlearning.org/

methodology used to teach this experimental course was SBL. SBL has been studied on its own in many research papers and have been proven to be beneficial as well as motivational. A study of only SBL as a factor would just be a repetition of such research. Furthermore, the SBL approach is a natural fit with the AIA tool and provides the educational setting and environment needed for such a language: this synergistic combination brings the advantages of active learning, peer learning, and motivational contexts to introductory CS. This approach was implemented on non-CS majors in order to find a sample that had little to no programming experience; a sample that may exhibit a negative attitude towards CS as a discipline or at least did not have one.

The purpose of this study was to implement an experimental introductory CS course for non-CS majors focusing on two pedagogic factors and to evaluate their effects on students' motivation, achievement, and attitude towards CS; evaluating them as a comprehensive learning environment. These factors were:

1- The use of a visual blocks programming language, in this case the App Inventor for Android (AIA); and
2- the adoption of Studio-Based Learning (SBL) as the main teaching methodology and using lectures, only when needed, to reinforce the SBL sessions.

The hypothesis was that these factors will have a positive effect on students' motivation, achievement, and attitude towards CS. Specific research questions were:

1- What effect will this approach have on achievement?

2- To what extent are aspects of this implementation found useful, relevant, and interesting by the students?
3- To what extent does collaboration, an integral part of SBL, relate to student success?
4- To what extent do the students relate given activities to creativity?
5- What relationship exists between this implementation and the comprehension of CS concepts?
6- What relationship exists between this implementation and attitudes towards CS?
7- What relationship exists between this implementation and students' comfort level?
8- What relationship exists between this implementation and motivation to learn in general?
9- What relationship exists between this implementation and the students' desire to learn CS concepts specifically?

2. LITERATURE REVIEW

Sloan and Troy [81] found that introductory CS courses were neither interesting nor engaging enough to attract students. Kurkovsky [50] argues that one of the reasons for low enrollment in CS is its decreasing appeal as a discipline or as a career. Today's students "need a classroom experience that they can relate to, that is creative and challenging, and makes a difference in preparing for their careers" (p. 44) [50]. Universities, specifically CS departments, are taking a second look at introductory CS courses and encouraging the implementation of new approaches. Many new approaches have been implemented with innovative teaching methodologies and programming languages and environments. This section takes a look at the problem of retention in CS, these new implementations and approaches, and the factors that may influence their success.

2.1 Retention

Many universities suffered from low retention and decreased enrollment in the CS major [81, 86]. An annual survey is implemented by the Higher Education Research Institute at the University of California at Los Angeles (HERI/UCLA) to study the

interest of incoming US undergraduates in different majors.[5] The survey showed that in 2006 interest in CS as a major has dropped more than 70% from its peak in the early 1980s [85] (see Figure 2-1). Introductory CS courses, moreover, tend to have high attrition rates and low success rates [18, 27, 81]. At University of Illinois at Chicago (UIC), Sloan and Troy [81] stated that the attrition rate among freshmen and sophomores majoring in CS was 40-60%. There, the problem was believed to lie in the introductory course sequence in CS.

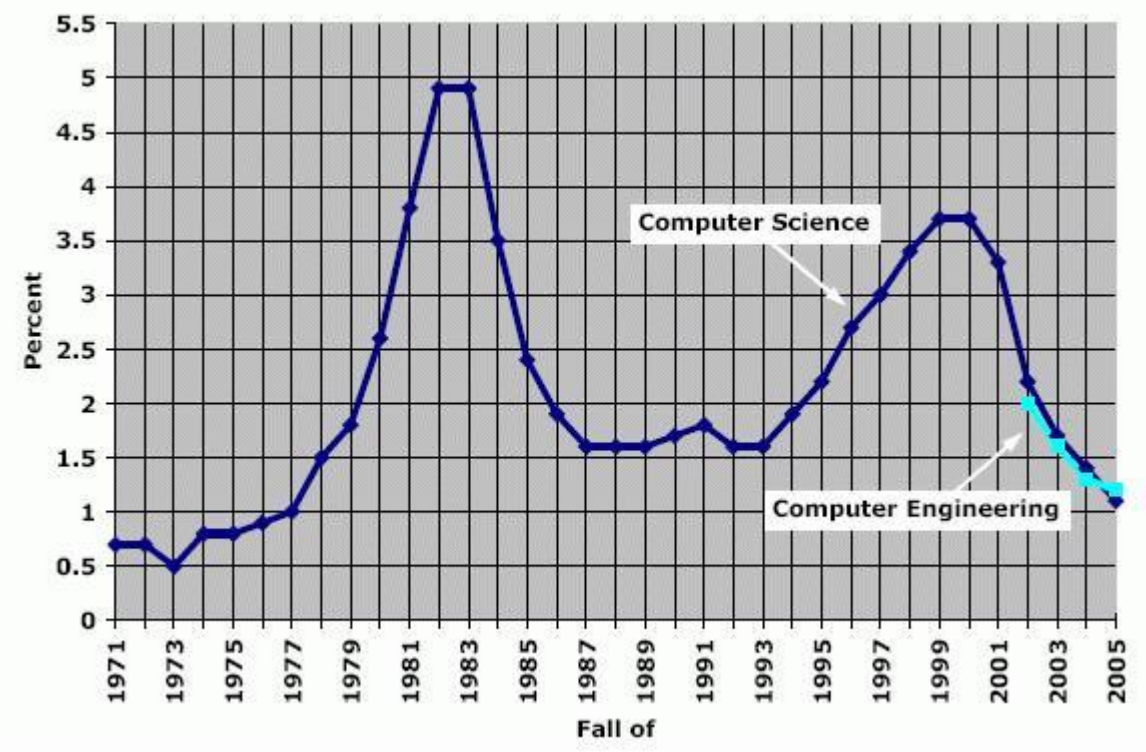


Figure 2-1: ***Interest in CS and CE as a Major Among Freshmen*** **[85].**

[5] For more information see http://www.heri.ucla.edu/

A study was performed by Lewis, Yasuhara, and Anderson [53] examining the factors that shape the students' decision to major in CS. To explore those factors, they gathered data through interviewing students attending CS1 and CS2 at two large universities. Thirty one students participated in this study; evenly split among genders. A third of these participants already intended to major in CS, another third were undecided or intending to minor in CS, and the remaining third were not intending to major nor minor in CS; the researchers did not mention whether this sampling was intentional or not, nor whether genders were evenly split within the three thirds. In their interview analysis, Lewis et al. recognized five factors influencing students' decision to major in CS, which were [53]:

- Their **ability** as related to CS: experiences and expectations of success as CS majors.
- Their **enjoyment** of CS: how much they would enjoy majoring in CS.
- The **fit** between their identity and CS: the extent to which their own values and identity align with values and cultural expectations they associate with CS.
- The **utility** of CS: the extent to which CS would provide potential value to society or to them as individuals.
- The **opportunity cost** associated with majoring in CS: practical constraints, as well as ways in which majoring in CS might restrict other plans.

Lewis et al. argued that students take these factors into consideration when deciding to major in CS. “Assuming that one role of introductory CS courses is to help students make informed decisions about majoring in CS” (p. 9), Lewis et al. claimed that CS educators can utilize the different information that can be gathered based on these factors to influence students. Educators can, furthermore, make decisions regarding CS courses, the learning environment, and even the CS curriculum using such information.

The retention rate of female students as well as minorities in CS is even lower [1,57,81]. Even though the percentage of female students in colleges (60%) is higher than males, very few of those who acquired a degree in computer science are female or minority students (<30%) [27]. Up to 2011, female students’ retention rate is still significantly lower than that of male students [91, 94]. Forte and Guzdial [27] emphasized the importance of turning that number around, raising the question that “if educated women are not learning to be computationally literate, what role will they play in a society whose forms of expression are increasingly defined by the computationally proficient?” (p. 2).

2.2 Factors Influencing Performance

There are factors and limitations that must be considered when innovating a course, such as hardware requirements and student background [83]. Tow et al. [83] stressed the importance of knowing the students and their interests and building on previous research. Introductory computer science courses include a diverse body of students, thus the traditional way of teaching the class no longer works and it needs to be changed to serve the needs of this diverse group in order for them to succeed [27].

Burgin and Reilly [14] studied the factors that may influence students' performance in programming in an introductory CS course. Those factors were previous academic experiences, cognitive abilities, and personal characteristics. Previous academic experiences with science subjects, especially math and science, were found to have a significant positive influence on performance [14, 88]. The researchers also studied previous programming and non-programming experiences [14]. They defined non-programming experiences to be experience with computer applications and games, using email, and browsing the Web. No significant differences were found between students with or without previous programming or non-programming experiences. In terms of personal characteristics, the comfort level students experience in the class environment has a positive correlation with their performance in terms of programming [14, 88], with male students having a higher correlation between comfort level and performance [14]. The strongest relationship found was the relationship between students' perception of their understanding and programming performance. No significant differences were found between males and females in previous academic experiences, cognitive abilities, and personal characteristics. These findings support the hypothesis that previous knowledge in math and science, comfort level, and perceived level of understanding of the concepts presented are all factors that positively influence students' success in an introductory CS course. The presence of these factors in an introductory CS course would assist in predicting success in achievement and performance of the students.

On the other hand, Rountree, Rountree, Rountree, and Hannah [76] argued that it is hard to predict achievement and performance in an introductory CS course. Establishing a comfort level that may predict success is difficult. This is due to the large

number of students in that course. Consequently, it may be hard to establish a comfort level that satisfies all students. Differences in high school level curricula make previous knowledge in math and science different from one student to another. A student simply having such previous knowledge may still not do well or may have a negative reaction toward the math content covered in the course. The diversity of the background skills, differences in motivation levels, and differences in expectations of the course are also factors that contribute to the difficulty of predicting achievement and performance. However, Roundtree et al. did find "the *desire* of students to pick up the skills of programming was the strongest influence on passing the course" (p. 102) and on the performance of the student. This is not news to those in the field of education. For Jerome Bruner, "famous for many visionary ideas in education" (p. 36) [26], has emphasized the importance of the desire to learn on the students' understanding and processing of knowledge [10]. But in an introductory CS course there are students who are taking the course who lack such desire. They may be taking it to complete their program requirements or to transfer from another program. Roundtree et al. [76] suggest motivating those students by explaining to them the required level of commitment and helping them aim for mastery of the skills rather than just earning a passing grade. They also suggested that "if it is possible to identify students who are 'at risk' early in the course, then it may be possible to target specific support to assist them toward success in CS1" (p. 104).

2.3 The Media Computation Approach

Previous research showed one promising solution, to introduce CS concepts through multimedia. This approach is not a new one; it goes back to the 1970s. Hanson's implementation in 1978 may have been the very first to try such an approach [36]. With the advancement of computer science and the current media technology, this approach seems to be more feasible.

Media computation provides relevant and challenging content where media is used to help facilitate the instruction in a relevant and interesting manner to the students [27, 33]. It allows students to be creative, and provides an unintimidating environment that encourages collaboration and increases the comfort level [27]. Furthermore, the approach supports learning as concrete-to-abstract, in the sense that most programming fundamentals are presented first in a simple manner and later are elaborated. Students are given the opportunity to apply these fundamentals in a concrete manner first, and later understand the abstraction behind them. However, problems arise for the educators "in creating environments that invite students to take advantage of these unfamiliar media in order to learn from them without first investing immoderate amounts of time learning to program the computer" (p. 2) [27].

Universities have implemented this media computation approach to seek solutions to the traditional introductory CS course. At Georgia Tech, Guzdial proposed a new course for introductory CS where "core computer science concepts can be introduced through media computation" (p. 104-105) [33]. It was designed to incorporate creativity, relevance, and collaboration [71]. This course followed the data-first approach, in which

students are first introduced to the encoding representation of the data and are then gradually introduced to algorithms and programming concepts as a useful way to manipulate that data [33,71]. The course however did cover many of the same contents as the standard CS fundamentals class [71]. A goal of this course was to enable students to develop "*tool building* skill[s], not *software development* skill[s]" (p. 105) [33]; to be able to understand and alter written program segments and use their functionalities, not just write code from scratch. This concept enforces many of the objectives in the *Computing Curriculum 2001* (CC2001) [2].

When the Media Computation course was first offered at Georgia Tech in Spring 2003, registration was restricted to non CS majors in an effort to reduce anxiety and intimidation, and create a hospitable environment as well as increase the comfort level [71].Throughout the course, students were encouraged to collaborate on homework assignments and participate in study groups. Students in the media course expressed great interest in the course and found it to be very relevant to them personally and professionally. Of the 120 students only two withdrew from the course, both male [33]. The results indicated that 63% of the students showed interest in taking another media computation course. Of the female students, 10% said they would take another computer science course, whereas 60% said they would take another media computation course. In terms of performance, the traditional CS course had an average rate of 27.8% of students who withdrew, failed, or got a D grade. The Media course had a better retention rate where only 11.5% of the students withdrew, failed, or got a D grade. However, these studies did not mention if any statistical tests were implemented to determine the significance of these results. In general, collaboration was found to be an integral part of

the course [27]. It led the students to better understand the material, it gave them an opportunity to discuss their problems and solutions with peers, which gave them confidence in their work, it provided them with an indirect way to get help, it created a relaxed atmosphere, and helped students feel comfortable in the classroom. Students were enjoying the material, "taking advantage of the creative aspects of the course and doing interesting things on their own" (p. 6) [27]. They found "programming media to be something fun and useful beyond the completion of assignments" (p. 6).

At Gainesville College, Tew et al. [83] applied the media computation course that was implemented at Georgia Tech as a CS0 course. It was taken by "students who did not feel that they had adequate background in computing to take the traditional CS1 course" (p. 417) [83]. Similar to Forte and Guzdial [27], the results showed students had improved programming skills and found the skills learned were relevant. Furthermore, 50% of the students stated they would take more courses in media computation compared to the 31% of students who were interested in taking additional CS courses. However, this success may have been attributed to the small class size of between 9 and 39 students, the in-class hands-on experience with the instructor available to provide help , the slow pace of the course progress, or to the fact that only half as much material was covered. Interestingly, results showed that students perceived media computation to be quite different from CS. Students were asked if they believed that "Media Computation teaches a different set of skills than other intro CS courses" (p. 420) and more than half of the students said yes.

At UIC, Sloan and Troy [81] hypothesized that introducing an introductory course before the traditional CS1 course would significantly improve retention and allow those

with better experience to skip the introductory course with no harm to their educational experience. Furthermore, entering CS1 with basic knowledge of programming will help the course progress smoothly and give students the knowledge and academic experience that would improve their chances of success in CS1 [14,81]. Similar to Tew et al. [83], Sloan and Troy [81] implemented the media computation course as an introductory course for CS majors. Some modifications were made to the Georgia Tech implementation's lecture slides and assignments. This course moved at a slightly slower pace with smaller required weekly "finger exercises," which were exercises that required 5 to 10 lines of code. Similar to Forte and Guzdial [27] and to Tew et al. [83], results of this study showed a high rate of success with 84.1% of the students getting a grade between A and C, that is, not withdrawing, failing, or getting a D grade [81]; compared to a previous rate of 75.9%. Retention also increased from 38% before the implementation of the course to 59% after. Retention was classified as taking the next required core CS course, which indicated that the student showed interest in continuing with the major. Students felt great improvement in their programming skills and found the material to be relevant to their lives and careers.

Freudenthal et al. [28] found Guzdial's media computational approach [33] to be created for use in Liberal Arts programs. In Fall 2007 they offered their own version of this approach under the name Media Propelled Computational Thinking (MPCT) that would be better suited for science related programs. Computational thinking "is being recognized as an important aspect of successful education, and traditional computer science units should be considered in terms of how they contribute to the overall field of computing" (p. 8) [19]. The Freudenthal et al. [28] version of the media comp approach

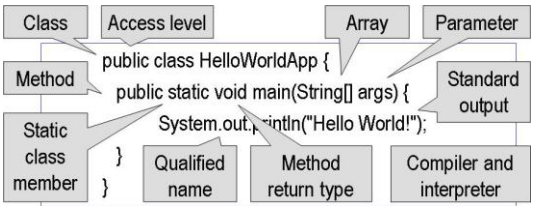
"uses image manipulation to strengthen mathematical intuition at the pre-calculus level, and to illustrate the modeling of physical processes" (p. 37). In this MPCT program, students would create graphical applications modeling physical phenomena using Python. Students would create programs that simulate bouncing of objects, harmonics, ballistic motion, etc, while learning basic programming concepts. Previous to the implementation of this course, success rates in CS1 ranged from 50% to 70%. After the implementation, however, "almost all students demonstrated proficiency at basic programming concepts and passed" (p. 41) the course. Freudenthal et al. found the MPCT course to be engaging to students "with weak math skills" (p. 41) regardless of their major.

2.4 Pedagogic Programming Languages

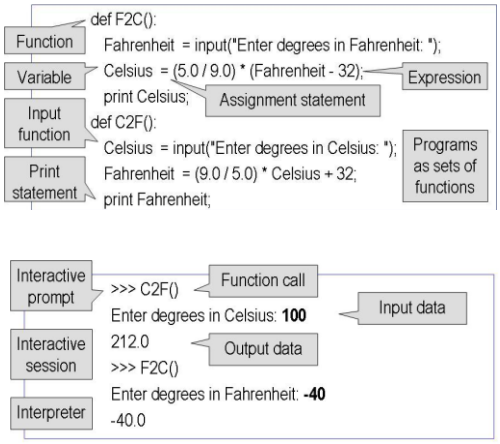
Introductory CS courses often use languages such as Java and C++ to introduce basic algorithmic concepts and fundamental programming constructs. However, the use of these languages in introductory CS causes students to struggle with the textual syntax [40]. Students spend more time trying to figure out how to deal with the development environment than on learning the fundamentals [40]. This becomes a problem when the objective of the introductory course is learning basic algorithmic concepts and fundamental programming constructs, where the use of such languages takes away from the importance of learning these basic concepts. Furthermore, the Computing Curriculum 2001 (CC2001) [2] encourages understanding of underlying algorithmic skills. Alongside learning the syntax of languages, Student should also recognize that they are effective problem solving tools, helping them recognize that a language has different attributes and tradeoffs with each given problem. Thus, as CC2001

commands, developing "their ability to adapt to different kinds of problems and problem-solving contexts in the future" (p. 23) [2].

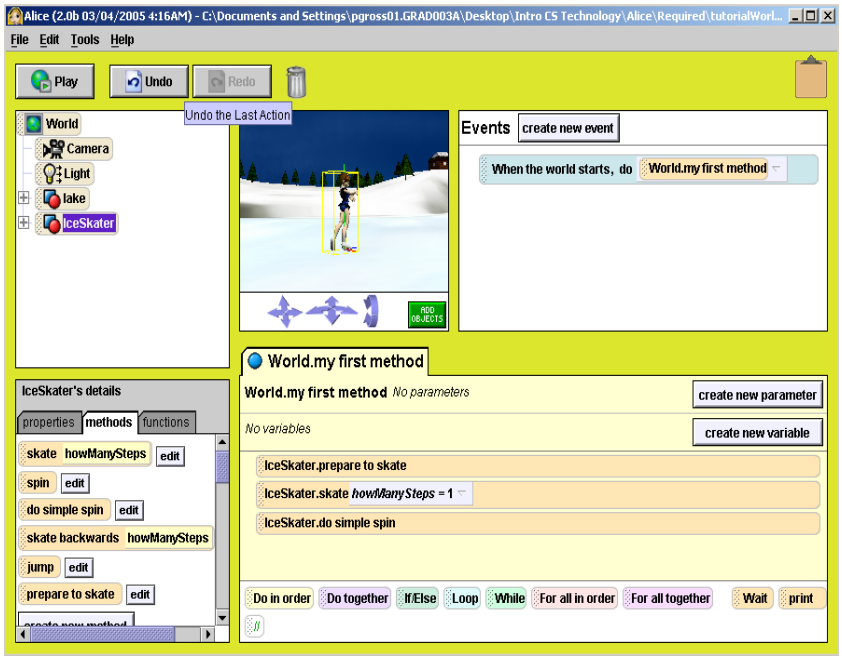
According to Hundhausen, Narayanan, and Crosby [40] "introductory computing education is too closely tied to programming languages" (p. 392). Recently, pedagogic languages and lightweight development environments are being used in introductory CS courses; languages such as Python, Alice, and Scratch [e.g. 69, 63, 89] and environments such as DrJava [e.g. 35]. These languages and environments are used as a gateway to transition to the *standard* professional-grade development languages and environments. In Python and DrJava students still have to learn textual syntax in order to create programs. On the other hand, in Alice and Scratch programs are created through assembling stacks of command blocks in a drag-and-drop environment using a mouse. They introduce an environment for the students that is syntactically less challenging (see Figures 2-2.a and 2-2.b for a comparison). As figures 2-2 illustrate, the simplest program created in Java needs an introduction to complex concepts such as classes and method declarations which is confusing for a novice programmer. Python on the other hand offers a simpler environment where students can dive right into creating functions, conditionals, and loops. Taking it one step further are the visual blocks languages. Using Alice and Scratch, students can create applications by dragging and dropping command blocks without having to worry about syntax errors, such as a missing semicolon. The blocks function like puzzle pieces, where command blocks are only allowed to be dropped in sequences that work correctly. The same is true of AIA.



Java [69]

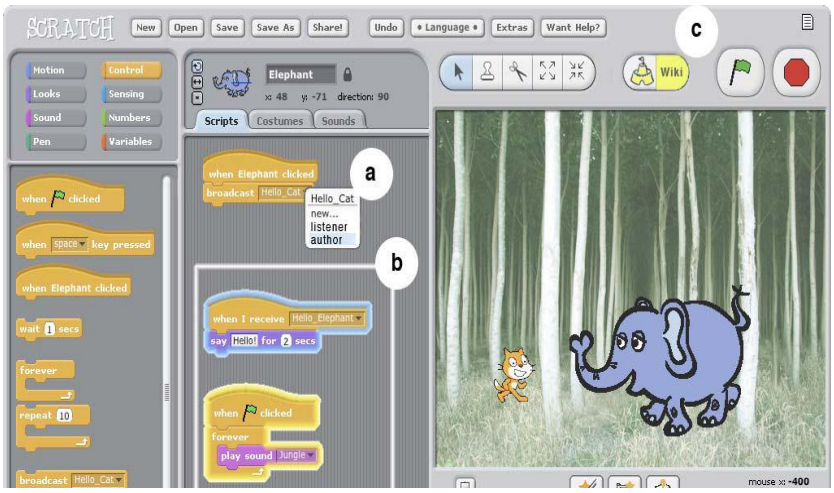


Python [69]

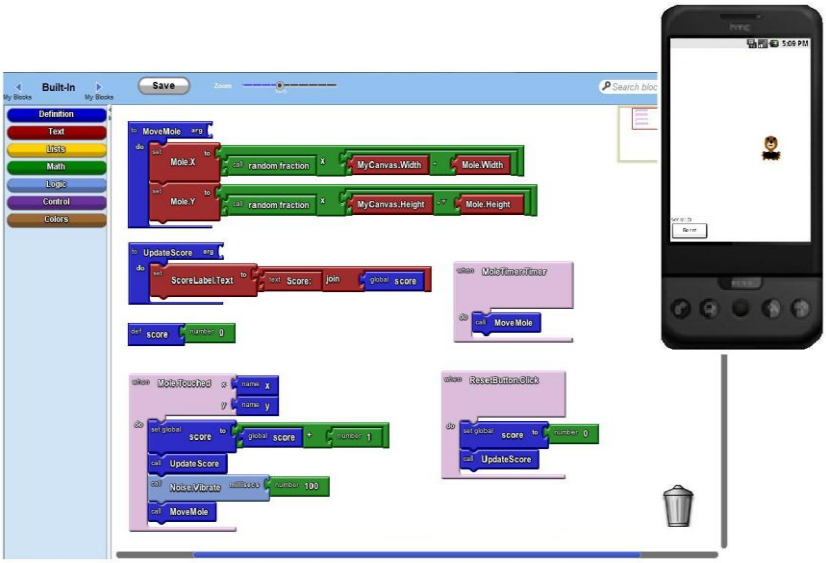


Alice [31]

Figure 2-2.a: ***Comparison of Programming Languages.***



Scratch [29]



AIA.

Figure 2-2.b: ***Comparison of Programming Languages.***

At Chapman University, a CS1 course using Python was implemented to be followed by a CS2 course using Java. The courses are required for computer science,

computer information systems, and mathematics majors. The researcher chose Python because of its simplicity, compared to Java, and because it offers a middle solution, that is it "was originally designed for education but soon gained [commercial] popularity" (p. 198) [69]. The CS1 course was lecture-based and included supervised labs. Of the students taking the course 53% were beginners. Throughout the course students were required to complete 37 lab assignments and were given 25 optional ones for extra credit. The researcher administered a survey after the completion of both CS1 and CS2. In the survey students were asked what activities helped their learning experience, they found the entire structure of the course to be the most beneficial, whereas lectures were the least beneficial. Furthermore, the survey indicated that students would strongly recommend Python as a first language for beginners. The researcher indicates that the course may contribute to lowering the attrition rates in CS since after its implementation the attrition rate in subsequent CS2 courses did not exceed 4%. The course was later approved by the university as a "general education science elective" (p. 197) [69] attesting to its success as an introductory CS course.

In Georgia Tech's Media Computations course [33], the language used to teach the course was a version of Python called Jython. Jython is an implementation of Python that runs on the Java Virtual Machine rather than C [33, 34, 44]. It combines the ease and flexibility of Python with the capabilities of Java, such as "servlets, database programming via JDBC, [and] GUI programming via Swing" (p. 106) [33]. Students were also provided with a set of Java and Jython classes that incorporate some multimedia functionalities needed to complete the exercises. A development environment, JES (Jython Environment for Students), was also created for the purposes

of this course. JES was a simple editor and program execution IDE. The reason behind this decision was to get the students started right away with the computation concepts without spending time on learning tedious and irrelevant coding schemes. This eases a concern Forte and Guzdial [27] had about the problem of spending much-needed time on learning tedious programming details when implementing a new approach. As mentioned earlier (section 2.3), the students had a better understanding of the material in this course and were less intimidated [33]. They even went beyond the requirements of the course and created interesting applications on their own.

At Slippery Rock University, a CS1 course was implemented using Alice 2.0 [63]. Alice 2.0 is an environment for creating 3D animations that does not rely on textual syntax. The researchers chose Alice due to its immediate rewards and the ability to directly manipulate created objects in its editor. The researchers found that the course took off an immense teaching load from subsequent CS courses. However, some students completed the course without feeling they had learned programming. The course was compared to its previous implementation using C++. The researchers found the number of students passing the course had increased by 4% and the number of withdrawals had decreased by 4%. Students were also performing better in subsequent courses, with an increase of 5% passing in CS2 and 8% passing in CS3. The researchers also found that retention rates had increased in subsequent courses, 11% in CS2 and 8% in CS3. They note that the number of students majoring in CS had decreased by 50%. However, enrollment in CS1 had increased by 10%, indicating its popularity among non-CS majors.

At Harvard University, an introductory CS course was implemented in which Scratch was used for the first few weeks [56]. After introducing students to fundamental

programming constructs using Scratch, they were transitioned into more complex concepts using Java. The purpose of this study was to improve the experience of first-time programmers [56]. The researchers surveyed the students throughout the semester to assess the effects of their approach. Of the 25 students that participated in the study, 52% had zero programming experience. Students' comments on the use of Scratch were positive. They found it to be fun, easy, and rewarding. At the end of the semester, students were asked "how their initial experience with Scratch affected their subsequent experience with Java" (p. 223) [56], 79% found it to have a positive effect. One student commented: "Scratch helped me get a general idea of how to think like a programmer" (p. 226). Another student commented on the transition to Java: "I was able to approach the first Java programs with an idea of how to tackle the problems" (p. 226-227). One student commented negatively on the experience stating: "its [*sic*] about 100 times harder than Scratch, and the results are much less enjoyable" (p. 227). Overall, the researchers found this approach to be a successful one since students found it exciting and it eased their transition into Java. The researchers found Scratch to be a viable "gateway to languages like Java" (p. 227) and a way to familiarize the students with the "fundamentals of programming without the distraction of syntax" (p. 227). The study, however, does not show the effect of using Scratch for a whole semester.

2.5 Mobile Application Development

Mahmoud and Dyer [54] argue that "most student activities focus on the social side of computing–instant messaging, e-mail, music, video, Internet and games rather than the basic applications" (p. 496). Thus, "there is a huge gap between academic

environments and how student[s] use their computers" (p. 497) [54]. Cell phones, iPod, and pocket personal computers are an integral part of today's reality [55]. Cell phones are no longer used just for voice communications; they have come a long way since their earliest creations. Today, cell phones run a variety of applications and are considered a necessity for people in general and students in particular [54, 50]. Kurkovsky [50] argues that to increase enrollment in CS and decrease attrition rates "CS curriculum [needs] to stay relevant to today's reality and engage students by making a strong connection between computing and their everyday lives" (p. 44) [50].

The use of mobile application development in the classroom provides students with relevance and practical experience which will inspire students, stimulate their interest, and motivate them to learn [54, 55, 50]. Skills in mobile application development are in high demand by the industry [50], and "the target mobile phone market is very accessible and extremely large with estimates of over two billion phones in use worldwide" (p. 45) [50]. Mahmoud and Dyer [55] claim that mobile application development "provide[s] a motivating framework for students to develop novel solutions for mobile applications. . . . [and renew] interest in pursuing a computer science major" (p. 108). However, there are challenges that arise in the design of such applications as well [24]. Mobile devices have limited input/output capabilities that must be taken into consideration in the design process. For example, the screen size is smaller than other devices and their pointing devices, if any, are harder to use while moving. Multitasking and task interruption support are key issues in the design of mobile applications due to their high frequency in mobile devices. It is important for students to

understand those limitations and their significant impact on the design and efficiency of the applications [50].

The difference in developing mobile applications is that those applications are developed on one platform then deployed to run on another [54, 55]. This presents an "opportunity to introduce students to different programming models" (p. 108) [55] as well as illustrates to the students the following:

> The logic of the application doesn't change no matter on which device it will run. It is the user interface and interaction model that changes and thus students learn about the different methods for reading input from the user and handling events. (p. 499)

There are many different platforms for mobile devices and developing mobile applications, such as BlackBerry, Microsoft Windows Mobile, Palm OS, Google Android, and Java Micro Edition (Java ME) [55]. Through these platforms, students develop applications using programming languages such as Java, C++, and Python.

At the University of Guelph and the University of Guelph-Humber, introductory CS courses were implemented in developing mobile applications [54]. The platform used was Java ME with specific configurations which were the Connected Limited Device Configuration (CLDC) and the Mobile Information Device Profile (MIDP). The mobile devices used were BlackBerry wireless devices. The course offered lectures and structured lab settings and focused on "problem solving, organizational approaches, and basic algorithms" (p. 108) [55]. Using Java ME in developing mobile applications was found to be challenging due to its limited library and functionalities. As a result, these functionalities had to be provided or students must develop their own. Also, a format

conversion process is required in order to run a created application on a BlackBerry device. After the conversion, the resulting file can be loaded onto a device using a USB cable. The results of this study were based on student feedback which indicated that they liked the experience and would like to see it implemented in other CS courses. The researchers found that students enjoyed learning about mobile application development "but also became aware of the development challenges they present" (p. 106) [55]. They found that the course gave the students real life examples and hands-on experience which raised their level of excitement and satisfaction with the course. They argued that the students' possession of the latest Java-enabled cell phones created "a motivating framework . . . and inspires them to work hard" (p. 107) [55].

At Central Connecticut State University, a mobile game development course was implemented for CS-majors [50]. Kurkovsky [50] argued that it is easier to adopt mobile game development than traditional game development due to the former's smaller scale, simpler graphics, and lesser complexity. The course was not an introductory one but was still positioned early in the CS curriculum. The goal of the course was to introduce advanced CS topics, such as data structures, artificial intelligence, and software engineering, through developing games for mobile devices. It required students to have experience in Java, that is, either completed CS1 or have Advanced Placement CS credit. Along with a variety of small assignments, a semester-long project was required to develop a playable game for mobile devices. Java 2 Platform Micro Edition (J2ME) was used along with Sun Java Wireless Toolkit (WTK) for testing, debugging, and deployment. The students' feedback confirmed that they found the course interesting, motivating, fulfilling, and enjoyable. The researchers found the design and

implementation of mobile game applications a feasible task for lower-level CS students. Furthermore, they found that "mobile applications and games offer instant gratification in the sense that students can download them to their mobile phones almost immediately and show them off to their friends" (p. 47) [50].

2.6 Studio-Based Learning

Studio-Based Learning (SBL) is a form of laboratory instruction that dates back to the middle ages [9, 60]. It is an instructional model that emphasizes active learning and has been used across many disciplines. In this model, students construct solutions to given problems, either in groups or individually, and present them in discussion sessions, known as *design crits*, where the instructor and fellow peers can give feedback. The model provides ample opportunities for individual reflection and social interaction. Hundhausen et al. [40] argue that through the *design crits* students would acquire important communication skills valuable for their future professions. They will have the "ability to present, rationalize, debug, and critique programmed solutions to design problems" (p. 394). The model is scalable to different class sizes, adaptable to meet the "local needs" of different courses, and technology-independent, enabling instructors to use any form of technology they are familiar and comfortable with [40]. Furthermore, SBL adheres to the constructivist approach which claims that "knowledge is actively constructed by the student, not passively absorbed from textbooks and lectures" (p. 257) [7]. According to Ben-Ari [7], "teaching techniques derived from the theory of constructivism are supposed to be more successful than traditional techniques" (p. 257). Therefore, SBL has the potential of being a successful instructional

model. Carter and Hundhausen [17] found "students responded favorably to studio-based activities" (p. 110). In fact, many studies implemented SBL with very positive results [e.g. 17, 23, 25, 42, 41, 38].

Many universities have recently implemented Studio-Based Learning. At the University of Illinois at Urbana-Champaign, a studio-based CS course was implemented for intermediate students [90]. In this course, each week students met in a one-hour lecture and a two-hour discussion section. They were given loosely specified programming assignments every two weeks and one final assignment that lasted for four weeks. The discussion sections consisted of five students and each student gives a 20-25 minute presentation of their work. Woodley and Kamin [90] found these discussion sections (aka *design crits*) to have several effects. They made students work harder "to avoid looking foolish in their presentation" (p. 533), made them write clear code in order to explain easily "which is a focus that nearly always produces better code" (p. 533), and depend on themselves thus preventing cheating. Discussion sections are led by graduate teaching assistants or undergraduates that did well in prior offerings of this course. What the researchers used as evidence of success was the students' testimonials that the course was helpful and the improvements in their coding as the course progressed. Other than that, this study did not present any concrete results.

In Australia, Monash University implemented Studio-Based Learning for a year-long core subject as part of an Information Technology (IT) program [16]. The aim of this core subject is to prepare students for their IT careers. The subject implements an integrated curriculum that "requires the students to use content and skills from other core subjects" (p. 214). This SBL model consisted of two studios, an Internet café, and a

meeting room. The studios were where most teamwork and group work took place. The internet café was used for informal meetings and socializing. The meeting room was designed as a professional space for consultations, meetings, and presentations. In this study, 132 first year students participated in a survey implemented during the last week of the first semester. The results of the survey indicated first year students' preference of this model over standard lectures, mostly due to its hands-on learning approach. The first year students indicated it was a positive learning experience and found the studio environment inviting and useful to their learning. These findings are encouraging, for they indicate that Studio-Based Learning can be successfully implemented in introductory courses.

At Washington State University, Hundhausen, Agrawal, Fairbrother, and Trevisan [41] wanted to implement SBL while still being able to give students individual programming assignments. To do so, they have developed what they called the pedagogical code review (PCR), "in which students first review each other's code solutions individually, and then come together in teams to identify, discuss and log issues with the code" (p. 500) [42]. PCRs were implemented in the lab sessions under the mediation of hired CS graduate students. Lectures, though, were still part of their approach. Hundhausen et al. [41] claimed their approach would provide students–on top of programming skills–with collaboration, communication, and critical thinking skills that are important to CS as a profession. They tested this approach in spring 2008 in a CS1 course teaching CS concepts using C as the programming language. Fourteen students participated in this study (12 male and 2 female). The PCRs were implemented three times during the semester. After each PCR an exit survey was administered. The

results indicated that students, by the end of the semester, seemed to focus more on the design issues rather than debugging and formatting issues. The students found collaboration beneficial and the quality of their work improved as the course progressed. Again in spring 2009, Hundhausen, Agrawal, Fairbrother, and Trevisan [42] repeated the study in the same CS1 course. In this study, however, the researchers compared their experimental approach to the traditional CS1 course taught in fall 2008 (i.e. without PCRs). In addition to the exit surveys implemented in the earlier study, a pre and post-test, a pre and post-survey using the Motivated Strategies for Learning Questionnaire (MSLQ) and the Sense of Community Questionnaire (SCQ), and an exit interview were implemented. A total of 176 students enrolled in both courses; 87 in the experimental and 89 in the traditional; of whom, only 42 students participated in the MSLQ and SCQ surveys, and 12 in the exit interview. An ANOVA was implemented on the data collected with no statistically significant differences found. Hundhausen et al., however, noted some "trends towards significance." In the results of the surveys, they found a significant decrease in the Self-Efficacy scale within the traditional course, and an increase in the Peer Learning scale in the experimental course compared to the traditional. In the tests, both the experimental and the traditional course showed a gain in performance. In the interviews, the results indicated that the students enjoyed both courses and the programming activities valuable to their learning. However, students attending the experimental course expressed a higher reception of positive feedback than those in the traditional and felt more comfortable with collaboration. With regards to PCRs, most students found them useful and helped build team practice.

At Auburn University, Hendrix, Myneni, Narayanan, and Ross [38] adapted SBL to a CS2 course on data structures in fall 2008. In addition to giving lectures, they implemented SBL in the lab sessions of the course where students would meet twice a week to work on projects in groups. The instructional language used in this course was Java. To evaluate their approach, the researchers administered a pre and post-test to evaluate mastery of CS concepts, as well as a pre and post-MSLQ and a pre and post-SCQ to evaluate attitudes. In comparing the pre and post-tests, the students showed an improvement in their learning as expected. In comparing the pre and post MSLQ and SCQ, the results showed a statistically significant increase in Intrinsic Motivation, Extrinsic Motivation, Self-Efficacy, Peer Learning, Effort Regulation, and SCQ scales. In spring 2009, the researchers collected data from the same CS2 course taught in the traditional format. They compared the post-test of the SBL course with the post-test from the traditional course, no statistically significant differences were found. However, when the grades of the students were correlated, a statistically significant increase was found in the students taking the SBL adapted course. This indicated that while both courses delivered sufficiently in terms of mastering CS basics, the SBL adapted course did "a better overall job at instruction" (p. 109) [17]. Hendrix et al. [38], however, did not implement MSLQ in the traditional course.

At the University of Victoria in Canada, Estey, Long, Gooch, and Gooch [25] implemented the SBL approach in a game design course. The purpose of their study was to promote collaboration and communication among students. Their approach, as well, was not purely SBL. The researchers still gave lectures as part of this course's teaching approach. Similar to previous studies, they used lectures to introduce new information

which were complemented by a SBL lab. Moodle was used as the course's management system; it was used to manage groups, create forums, and give online quizzes. Flash was the tool used to develop the students' game projects. The researchers spent the first few weeks introducing the students to Flash and Actionscript through a series of tutorials. The students were required to complete two major projects in this course, one in the middle of the semester and one near the end. In both projects students created games from scratch over five milestones: a game concept, three game prototypes, and a practice presentation. Each milestone involved a peer review process. At the end the semester, the researchers implemented a survey in which students would respond to statements according to a five point likert scale. The results indicated that students responded positively to the course. The students indicated an improvement in motivation and a benefit from the different perspectives and the sense of community SBL provided.

Gottel and Schild [30] also studied a method for cultivating creativity in a game design course. Their method was much like the SBL methodology. The purpose of their study was to "remind participants to focus on creativity and play-testing instead of on underlying technical or programming issues" (p. 98). In this game design course, students were divided into three groups where each group is required to create a game. The course took place in a room the researchers called the Creativity Room 5555. Similar to *design crits*, the purpose of this room was to provide a space for group members to meet and discuss their projects in a PC free social environment. Students, furthermore, were to use this room to create prototypes of their games and present it in four milestones. In addition, the researchers "booked two traditional PC labs" (p. 99) however none of the students made use of them; students often brought their laptops into the Creativity Room.

The researchers often met with the students in this room and introduced them to different topics such as brainstorming techniques. The researchers observed that the students enjoyed "the idea of having a room as [a] creativity environment" (p. 101). Gottel and Schild found "a strong need for creative and playful meeting places, especially in game design context" (p. 101). They claim that "even though it is hard measuring creativity, it was obvious that the resulting games contain unconventional features that contribute to attractive gameplay" (p. 101). The researchers found the game designs to be "well thought out and structured, extensively reviewed and focused on social interaction" (p. 102). However, they claimed that "team processes and planning decisions were poorly visible in 5555" (p. 102). Alongside their observations, the researchers conducted three surveys at the end of the semester. Two surveys were conducted on the students taking the course and one was conducted on visitors that attended the final demos of the students' projects. The results of the surveys indicated that the visitors "recognized creative elements in the games" (p. 101). They indicated, as well, that "the students appreciated the informal and pleasing atmosphere that let them exchange with the other participants" (p. 101). In general, the study showed the Creativity Room 5555 to "play an active part" (p. 101) in the process of game design and development in this course.

2.7 Summary

Low enrollment and retention rates in CS have brought about a reexamination of the discipline itself. The problem seems to be with students that are at the beginning of the major and with the general attitudes towards the major. Introductory CS courses have

been reexamined for performance factors that may contribute to a better implementation. Those factors were current students' interests, previous academic experiences, perceived level of understanding, comfort level, and the desire to learn CS.

Using media in a context that is familiar to the students makes the learning process more meaningful and raises their interest. Georgia Tech's implementation of the media computation approach combined using multimedia with the use of a pedagogic language [33]. The students showed an increase in the desire to learn the material compared with the traditional introductory CS course, even though the media course covered similar material. They showed higher interest in the discipline and a better level of understanding. They also found relevancy in the material being taught.

The use of pedagogic languages has provided a flexible and easier programming environment for novice programmers. It has had successful implementations in introductory CS courses, improving retention as well as success rates in those courses. More importantly it has helped prepare the students for the next level of CS courses, thus lowering the teaching load. Their use was even found to facilitate transition to the more complex professional-grade languages.

Searching through the literature shows few studies on the benefits of using mobile application development in introductory CS courses, but those that do exist have shown its positive effect in CS courses. They provide relevancy and motivation that students often seek in their learning experiences. Mobile application development platforms and environments are available but they are complicated to use in an introductory CS course. AIA provides a less complicated environment, however it comes with some restrictions;

the mobile devices that run applications created using AIA must have the Google Android platform.

Studio-Based Learning offers a socially active environment for students combating the anti-social image CS has as a discipline. It also provides an environment for reflection and acquiring much needed communication skills. The studies that implemented SBL show its potential to be a successful approach in introductory CS courses. However, there seems to be a senses that "SBL is appropriate for reinforcing ideas already learned but not for introducing new ones" (p. 110) [17]. Hence, there is a need of further research on the use of SBL as the main teaching methodology and for the *introduction* of new ideas.

The various new approaches and implementations showed high performance reflected in students' grades and seemed to be successful. Their success appears to be due to a mix of collaboration, medium used, relevancy, and performance factors. One cannot point to one approach and clearly say that it was *the* successful one.

3. METHODS

In this study, an experimental approach was implemented in an introductory CS course for non-CS majors. The course integrated the use of SBL with a new visual blocks programming language known as AIA. Data was collected using the Motivated Strategies for Learning Questionnaire (MSLQ), Interviews, mind maps, student course work, and observations. Data was also collected from a traditional introductory CS Course using MSLQ. The data was studied in order to evaluate the effectiveness of the applied methodologies.

3.1 An Experimental Course

The experimental introductory CS course was implemented in CS116 - *Visual Programming* in Fall 2009 at Ball State University (BSU). CS116 is a three credit hours introductory CS course for non-CS majors offered at BSU. It is a terminal programming course for non-CS majors, unlike CS1 which purposely introduces programming topics to prepare CS majors and lead them into CS2.

This experimental introductory course, CS116, was instructed by Dr Paul Gestwicki and Khuloud Ahmad. A course syllabus (see Appendix A) and a well devised plan of instruction implementing AIA and SBL were created for the course. The syllabus

included a description of the course, the prerequisites, the course objectives, the content covered in the course, and the format of the course in terms of learning environment and assignments. The course objectives cover the important learning aspects students must take from an introductory CS course while using all levels of the Bloom's taxonomy (see Appendix A), thus providing a more comprehensive learning experience. The main objective of this course was for students to think algorithmically rather than syntactically and be more competent in computational problem solving.

The format of the course involved learning basic algorithmic and programming concepts through completing project-based programming assignments. AIA was the instructional language chosen for this course. AIA was used in the implementation of the programming assignments creating applications for G1 smart phones (see Figure 3-1). The course was taught in a lab that included work stations equipped and ready for use. Each student had his/her own workstation to use during class meetings. Whenever the instructors felt students were unable to complete a given assignment on their own, or that they needed to comprehend a concept in order to reach a solution, mini lectures were given with hands-on tutorials during class sessions. These mini lectures were to simply reinforce concepts covered during SBL class sessions. In other words, the need for and the topics covered in the lectures were driven by the studio experiences. A total of five mini lectures were given throughout the semester. The specifics of the study, including the course format and the assessment plan, are described below.

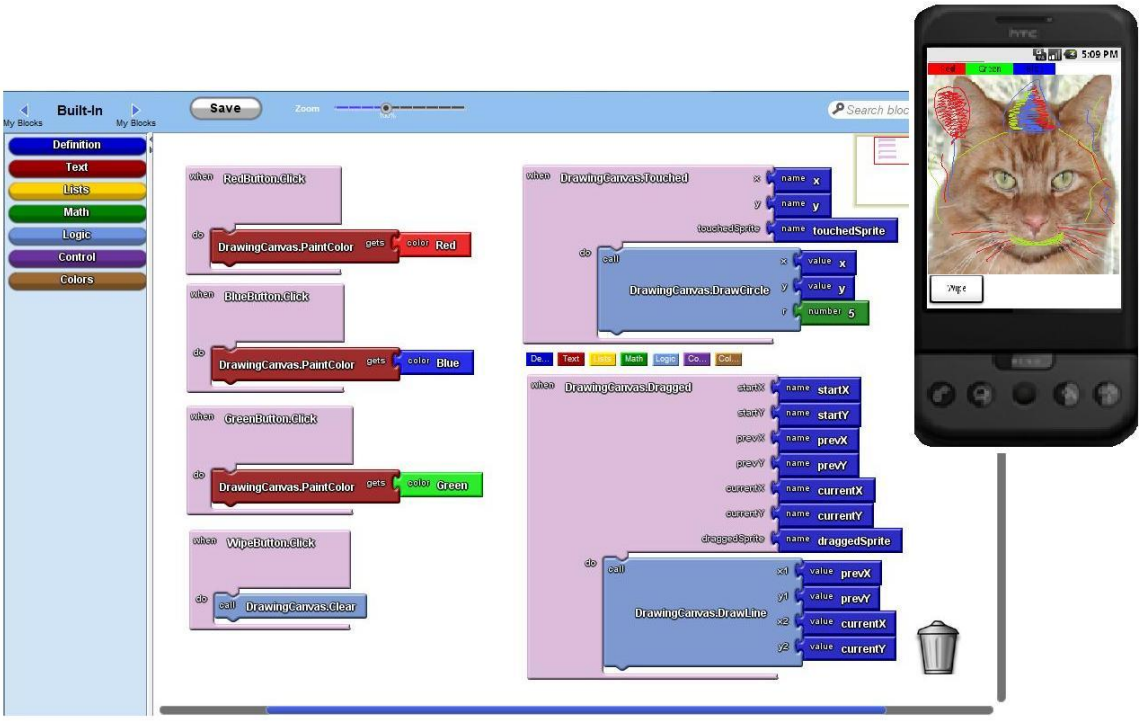


Figure 3-1: ***Paint Application Created Using AIA.***

3.1.1 The Structure of the Course

Group page. To facilitate communication and collaboration between instructors and students outside of the class, a website was created for the course. The site needed to be restricted to CS116 students only. It also needed to allow students to post their assignments and comments and ask any questions they may have. A group page was created using Google Groups (Figure 3-2), since students needed to create Google accounts to access AIA anyway. This was a perfect fit for this study for it complements the SBL methodology in its ability to facilitate communication and collaboration not only between instructor and student but between the students themselves.

Figure 3-2: ***CS116 Group webpage***

The instructors, also, used the group page to post assignments, reminders, reading materials, and comment on the students' work. The students used the page to upload their assignments and comment on their classmates' work as well. This helped in promoting active learning and student communications, two integral parts of SBL. Several of the features, however, that were relied upon during this study were later removed from Google groups and are no longer implemented.

Programming assignments. The first week of the semester was spent on introducing the students to the structure of the course (see Table 3-1). For the next few

weeks, students were given the assignment of completing the tutorials provided on the official AIA site [5]. This was to provide the students with the chance to familiarize themselves with the AIA IDE and the G1 phone as well. Throughout the rest of the semester, students completed a total of eight projects; six one week programming assignments, one midterm project, and one final project. Furthermore, of these six programming assignments two were completed individually and four in groups. In group programming assignments, groups were formed by the instructors using several methods: by major, by gender, by GPA, or simply by random selection.

Content	**Week**
Introduction to AIA	1
Complete AIA tutorials	2-4
Complete five programming assignments	5-8
Complete the midterm project	9-10
Complete a programming assignment	11
Complete the final project	12-15
Presentations of final projects	16

Table 3-1: *Course Structure.*

Following the SBL model, each project was discussed in the *design crits* on three progressive stages: *The Pitch*, *The Studio*, and *The Presentation* (see Figure 3-3). *The Pitch* starts before any programming is actually done, during which students were given

15 minutes to formulate their ideas and design their project. Some assignments were given themes to follow, for example service-related applications or education-related applications. This was done in order to foster creativity and give students some guidance. The instructors, moreover, were concerned that students would limit their projects to games or what they know how to do. After creating their initial design, students would then "pitch" their design to the rest of the class and get feedback. They would also get guidance from their instructors throughout the process. For the final project, students were also asked to submit the following in writing:

- A title for their application
- A mission statement that included the identified need for their application idea, the considerations of usability, and the intended audience for their application's concept
- The blocks intended for use
- Their rationale for the choice of application and design.
- The marketability of their application after completion

During *The Studio*, students would spend time in class working on their projects. This allowed the implementation of active learning. When obstacles were faced, students were encouraged to open a discussion session with the entire class for help in finding a solution. As during *The Pitch*, the instructors would also provide guidance while students attempt to understand components of AIA and how to implement concepts learned in their projects. In the final project, students were expected to incorporate most of the concepts they learned and the more complicated components of AIA. Therefore, they were given four weeks to submit their final product. For large-scaled projects, one or

more status reports were presented where each student discussed his/her progress and received feedback during class session. *The Presentation* is the last stage of the projects where students would "present" their finished, in rare cases close to finished, projects to the entire class. They would demonstrate how their applications work then explain the structure of their projects in terms of blocks sequence and design layout. Similar to the previous stages, presenters would then open the floor to discussion and get feedback from the rest of the class and the instructors. This feedback was intended to help students in their upcoming projects.

Figure 3-3: ***Stages of a Programming Project***

Reflection papers. After each programming assignment, students were asked to write a reflection paper; in which they were to write a self-evaluation in a minimum of 150 words. This evaluation would include a reflection on their contributions on the assignment, their strengths and weaknesses, the difficulties they faced, the issues they faced and what they found to be effective approaches to solving them, and their personal goals for improvement. There were two purposes to this assignment. One was to get a deeper insight into what the students faced during class and in working on their assignments. The other was to foster metacognition and reflective practice.

Formative Assessment. Essential to the SBL model is the process of formative assessment, both formal and informal. It is the bases for the active learning process that is intrinsic to the structure of SBL [17]. Therefore, the structure of the course was developed to incorporate formative assessment. This is most evident in the structure of the programming assignment and their division into the three stages: *The Studio, The Pitch, and The Presentation*. At each stage, the instructors gave students immediate feedback on their efforts. They, moreover, took careful notes and observations of the students' work. These notes were taken into consideration when grading each assignment.

Summative Assessment. One of the challenges faced in structuring this experimental introductory CS course was summative assessment. Students often worked in groups and, if not, solicited the help of others in their work. Therefore it was not fair to use submitted programming assignments alone for the purposes of summative assessment. To provide fairness in grading, it was imperative to find an additional way of assessing the student's individual achievement and assigning grades. For that reason, an individual midterm assessment process was devised in an interview setting in lieu of a written exam. Students would schedule an individual meeting with the instructors outside of the class for this assessment-by-interview.

The assessment-by-interview process included two parts. The first part was the sample application test. The second was the submitted midterm project. In the sample application test, a medium-size example application was created and students were asked to read and explain its functionality during the individual meeting. The aim was to test

their ability to distinguish and understand specific programming concepts that were taught in class up to that point, those included:

- sequencing of blocks: that is, the idea that execution flows sequentially when blocks are chained together vertically
- event-driven programming: responding to user and sensor events using blocks
- selection: the use of "if" and "ifelse" blocks to control execution of the program
- arithmetic: using blocks to do simple mathematical operations and random number generation
- variables: strings and string operations such as concatenation, lists, and the use of a shared index

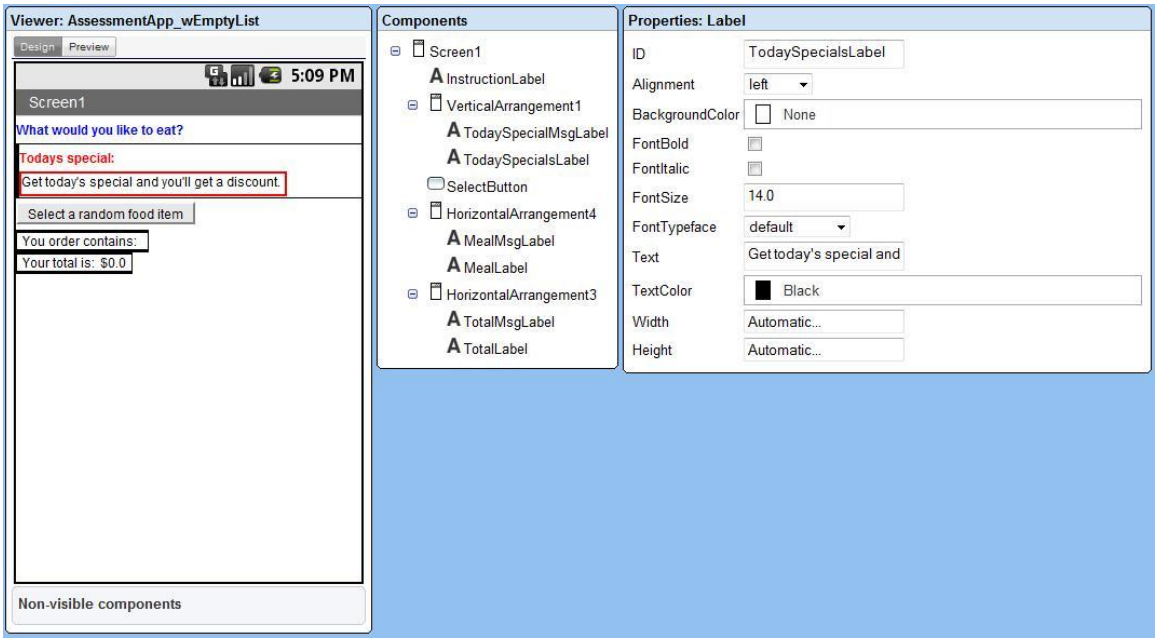
The application created for this purpose was a menu app: one list has the foods, another list has the prices. The application would randomly pick two things from the menu, add the prices, and show the cost with some built-in specials, for example picking rice & anything else gives you $0.50 off (see Figure 3-4 for application).

Along with understanding of the specific programming concepts mentioned above, students were expected to be able to do the following:

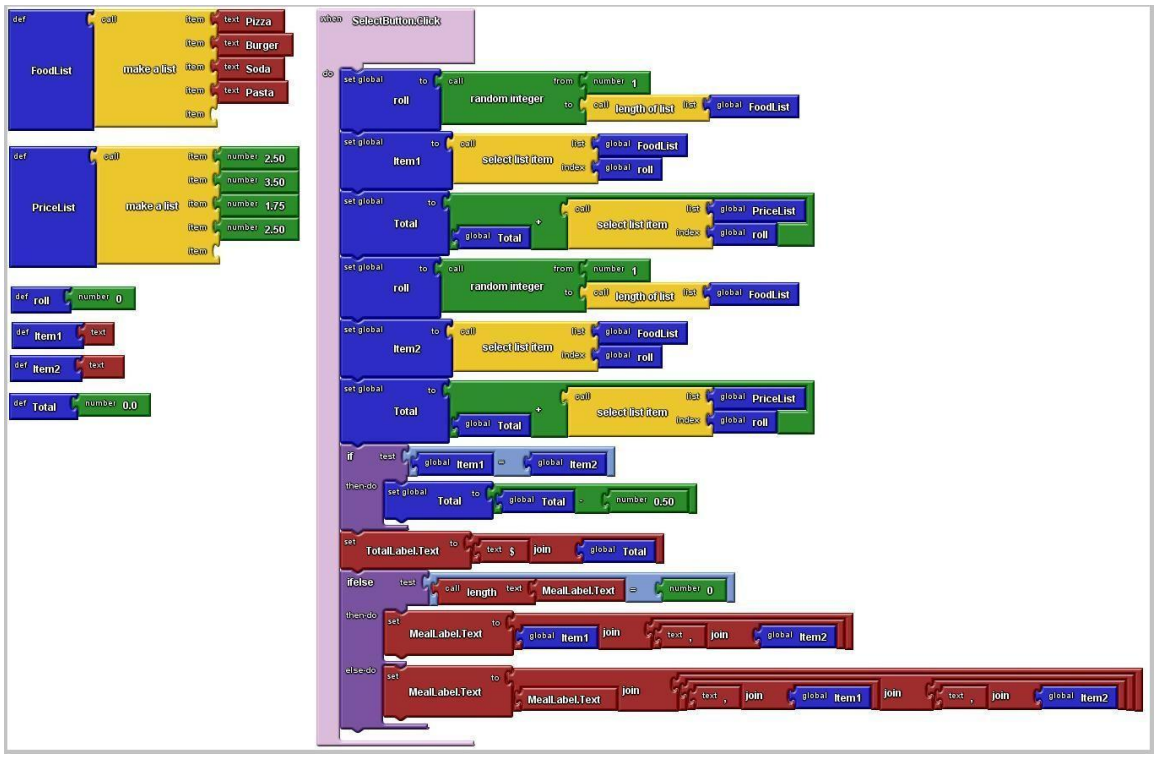
- understand loops: how to repeat sections of code
- manage multimedia assets such as sound and images
- show understanding of functional decomposition: the use of procedures and functions to break down an application into smaller more manageable pieces
- have developed elementary debugging skills
- know their way around the design view and the blocks editor

Those aspects were assessed using the midterm individual project that was included as the second part of this assessment-by-interview process. Each student was asked to submit a screenshot of the design view and the blocks editor, and a personal reflection paper on their project before their individual meeting time. During the assessment-by-interview, students were asked questions regarding their projects and were evaluated accordingly.

At the end of the semester, a final programming project was assessed as part of the students' final grade. In that project, students were asked to create an application using the more complex components in AIA. This was a four week project. The students were expected to deliver a more complex application that showcased their understanding of the more complex concepts in CS covered during the course. Similar to the midterm project, each student was asked to submit a screenshot of the design view and the blocks editor, and a personal reflection paper on their final project.



a. Application design



b. Visual blocks

Figure 3-4: ***Example Application Used in Midterm Assessment.***

3.2 A Traditional Course

Data were also collected from a traditional lecture-based CS course, CS110 - Introduction to Computer Science. CS110 is a three credit hours introductory CS course. Similar to the experimental course, CS110 is open to non CS-majors only. No programming experience is required however sufficient understanding of high-school mathematics is. The course provides a first look at the study of computing, including its history, architecture, programming languages, and application development. Students are expected to use and create programs. CS110, in contrast to the experimental course, has a co-requisite which was a one hour long lab session taught once a week in a computer lab. In these lab sessions, students created applications implementing what they have learned in the lectures. Only the MSLQ was administered in this course. The data collected from CS110 were compared with those collected from the experimental course.

3.3 Recruitment of Participants

Participants of this study were undergraduate non-CS major students registered in CS116 - *Visual Programming* section 001 and CS110 - Introduction to Computer Science section 001 in Fall 2009. Participants were chosen to be non-CS major in order to ensure they are novice programmers with little to no programming experience. In order to participate in the study, participating students were required to sign consent forms (see Appendix B and C). Participation, however, was voluntary.

Registration in CS116 was limited to 18 seats due to the limited number of G1 phones. By the end of the registration period, the total number of students that registered

in the course was 18. Students in CS116 were invited during the first lecture to participate in the study. After introductions, the recruitment script for CS116 (see Appendix D) was read followed by answering questions the students had. Consent forms were distributed to the students by a graduate student, other than the author. Students were asked to sign the forms and then place them in an envelope. The envelope was then sealed and taken to the Computer Science Department office for storage in a secure filing cabinet until the end of the semester. All 18 students registered in CS116 signed a consent form (3 female, 15 male). However, one male student withdrew from the course midway through the semester making the total number of participants from this course to be 17(3 female, 14 male).

In CS110, registration was limited to 45 seats and 42 students were registered. In the lab, however, registration was limited to 25 in order to provide each student with his own workstation. This resulted in having two lab sections available for students of CS110 to register in. Only one lab section was chosen for recruitment in this study. Students in CS110, similarly, were invited during the second week of the semester to participate in the study. The recruitment script for CS110 (see Appendix E) was read at the beginning of the class followed by answering questions the students had. Consent forms were distributed to the students and the instructor left the classroom. The researcher asked those who agree to participate to sign the form and then collected and placed them in an envelope. The envelope was then sealed and taken to the Computer Science Department office for storage in a secure filing cabinet until the end of the semester. Of the students registered in CS110's lab section, only 12 participated in this study (8 male and 4 female).

Overall, a total of 30 students agreed to participate in the study. Eighteen of them were CS116 students and 12 were students in CS110. The steps followed throughout this study to collect data were put in place to protect the human subjects, as required by the Institutional Review Board (IRB) for approval of the study.

3.4 Data Collection Tools

3.4.1 Motivated Strategies for Learning Questionnaire

Variations on the Motivated Strategies for Learning Questionnaire (MSLQ) [68] were administered in this study. The MSLQ is a "self-report instrument designed to assess college students' motivational orientations and their use of different learning strategies for a college course" (p. 3) [68]. It rates the participant on 15 different scales, they are the following [68]:

1- Intrinsic Motivation: this scale measures "the student's perception of the reasons why" (p. 9) he or she is engaging in the learning process being internal reasons "such as challenge, curiosity, [and] mastery" (p. 9). A high rank on this scale shows an interest in the course itself.

2- Extrinsic Motivation: this scale measures "the student's perception of the reasons why" (p. 9) he or she is engaging in the learning process being external reasons "such as grades, rewards, performance, evaluation by others, and competition" (p. 10).

3- Task Value: this scale rates the students' evaluation of the importance and the usefulness they see in what they are learning. A higher rating on this scale

refers to a need to be more involved in one's own learning; that is seeing a value in the learning gained through the course "in terms of interest, importance, and utility" (p. 11).

4- Control of Learning Beliefs: this scale "refers to students' beliefs that their efforts to learn will result in positive outcomes" (p. 12) in their academic performance. This scale focuses of the students own efforts rather than "external factors such as the teacher" (p. 12).

5- Self-Efficacy for Learning and Performance: this scale measures the students' perception of their own ability to master the learning material which "includes judgments about one's ability to accomplish a task as well as one's confidence in one's skill to perform that task" (p. 13).

6- Test Anxiety: this scale refers to the rate at which the students' mental and emotional states affect their performance.

7- Rehearsal: this scale measures the type of basic rehearsal strategies implemented by the student and their influence on his or her learning.

8- Elaboration: this scale rates the students' use of elaboration strategies in their learning, which "help students store information into long-term memory by building internal connections" (p. 20).

9- Organization: this scale rates the students' use of organization skills in their learning process.

10- Critical Thinking: this scale reports the rate at which students use critical thinking and apply previous knowledge to making decisions within their learning process or to solving new problems.

11- Metacognitive Self-Regulation: this scale rates how students check and correct their behavior as they learn. It focuses on "the control and self-regulation aspect of metacognition" (p. 23) through planning, monitoring, and regulating activities. Such activities "assist the learner in understanding the material and integrating it with prior knowledge" (p. 23) and in the "adjustment of one's cognitive activities" (p. 23).

12- Time and Study Environment: this scale test students' ability in time management and in regulating their study environment. This involves the activities of planning, the effective use of the study time, and the setting of realistic goals.

13- Effort Regulation: this scale measures the rate at which students' are committed to completing their study goals "even when there are difficulties or distractions" (p. 27).

14- Peer Learning: this scale rates the students' desire to collaborate and engage in dialogue with peers throughout the learning process.

15- Help Seeking: this scale measures the rate at which students seek assistance when needed either from their peers or from instructors.

These scales are divided into two groups and into several components based on their relationship to each other (see Figure 3-5). A detailed description of the questionnaire and the reliability and validity measures for each scale in the MSLQ can be found in *A Manual for the Use of the Motivated Strategies for Learning Questionnaire* [68].

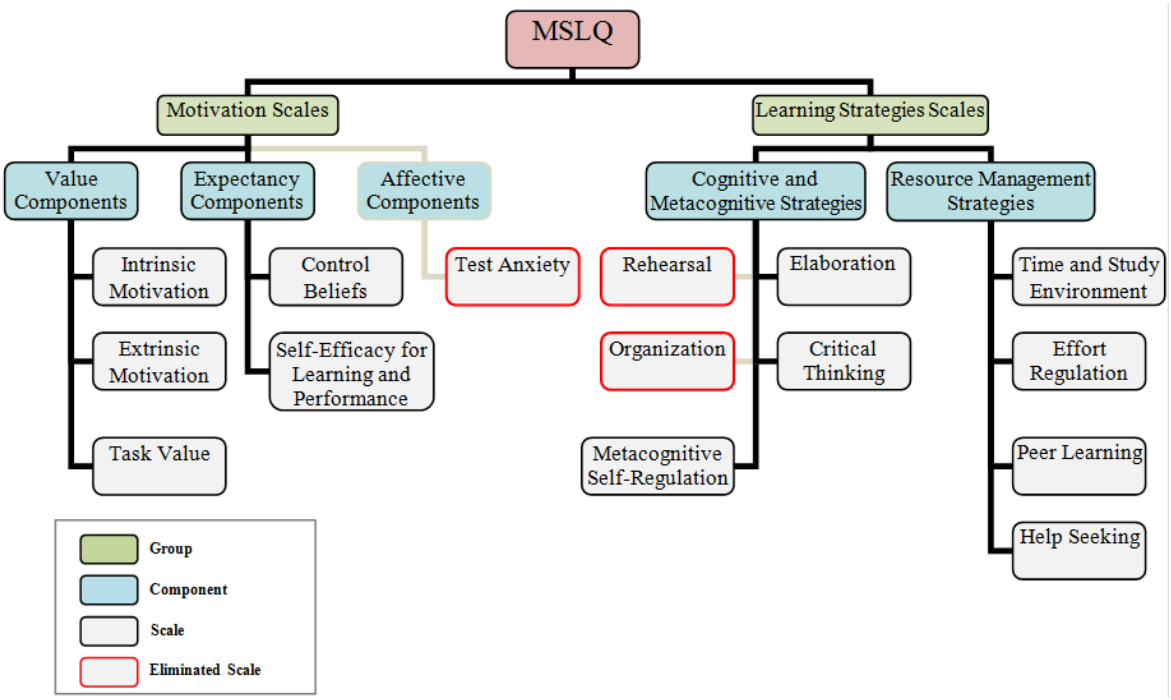


Figure 3-5: ***MSLQ Scales.***

The questionnaire contains a total of 81 items, each belonging to one of the scales above. Each item is a statement describing learning strategies, studying skills, or motivation for and attitude about a college course. For each item, participants indicate how true the statements are of them on a 7-point scale, with 1 being "not at all true" and 7 being "very true."

In this study, the original MSLQ was modified slightly and three variants were developed, an entry-MSLQ, a mid-MSLQ, and an exit-MSLQ. The modifications included the elimination of scales in all three modified MSLQs that were not relevant to the purpose of the study, thus the items related to those scales were also eliminated. The scales that were eliminated were Test Anxiety, Rehearsal, and Organization (see

Figure 3-5). In the Background section, in all three modified MSLQs, a few questions eliciting some background information were added. In the entry-MSLQ, some questions were changed to future tense to match the chronological order of the administration of the questionnaire.

After scale eliminations, 68 items remained in the modified MSLQ out of the 81 items in the original. The modified MSLQ also requested background data and study habits. This data was elicited through several questions stated in the Background Information section. This section also inquired about participants' study habits and attitudes about science subjects in general and CS subjects in particular. The three modified MSLQs are included in the appendixes (Appendix F, G, and H).

The modified MSLQs were administered during class time. The paper-based questionnaires were collected and sealed by a student volunteer, who took them to the Computer Science Department office for storage in a secure filing cabinet until the end of the semester. In CS116 the entry-MSLQ was administered at the beginning of the semester, the mid-MSLQ was administered at a mid point during the semester, and the exit-MSLQ was at the end of the semester. In CS110 only two of the modified MSLQs were administered, the entry-MSLQ and the exit-MSLQ. The entry-MSLQ was administered at the beginning of the semester and the exit-MSLQ was administered at the end of the semester.

At the end of the semester and after grades had been submitted, the questionnaires were scored. An electronic record of these scores refers to students only by arbitrary labels to protect anonymity. The electronic records were kept on the principal instructor's

workstation with backups on secure departmental servers, and the original paper records were destroyed.

3.4.2 Interviews

Two interviews were conducted with student volunteers from CS116; an entry interview at the beginning of the semester, and an exit interview at the end. Both interviews included questions that elicited responses about the participant's attitude towards this CS course in particular, towards CS as a major, as well as towards active learning and SBL (for lists of the questions see appendixes I and J). The exit interview, however, included in addition a few questions about AIA eliciting their attitudes toward it as a learning language. As an incentive, students were offered a $5 Barnes & Noble gift card for each interview. No interviews were conducted on students enrolled in CS110.

The entry interview was conducted by a graduate student who was not a stakeholder in the success of the project. It was recorded and kept in a secure filing cabinet until the end of the semester. Students who chose to participate scheduled interview times directly with the graduate student via email. After the semester ended and grades were submitted, the interview was transcribed and coded prior to analysis. The interviewee's name was replaced by an arbitrary label to protect anonymity. Note that the transcription did not take place until grades were assigned, so that student's responses cannot affect his grades. The original digital records were deleted after the transcription process.

Out of the 18 students registered in CS116, only one student volunteered for an entry interview. Therefore, to elicit more responses for the exit interview after witnessing

the extremely low responses to the entry interview, a change in the interview form was needed. An online form, that included the exit interview questions, was created where participants can go online and answer them. The form was created using Google Docs. Choosing Google Docs was based on the fact that responses are stored neatly in a spreadsheet online. Near the end of the semester, the students were asked to access the form and answer the questions listed. The researcher explained to the students that participation is voluntary and that the cutoff date was the grade's due date. To preserve anonymity, no names were required on the form.

3.4.3 Mind maps

One critical assignment that was included to analyze students' understanding of CS was the creation of mind maps with reference to computer science. A mind map is "a diagram that shows topics and how they are connected" (p. 171) [43] (see Figure 3-6 for example). It is a technique for enhancing creativity and productivity developed by Tony Buzan with Barry Buzan in *The Mind Map Book: How to Use Radiant Thinking to Maximize Your Brain's Untapped Potential* [15]. According to Buzan et. al, "Radiant Thinking reflects your internal structure and processes. The Mind Map is your external mirror of your own Radiant Thinking and allows you access into this vast thinking powerhouse" (p. 31). They suggest that mind maps offer away to represent the internal mental image on an external piece of paper which would allow the chance for analysis and interpretation.

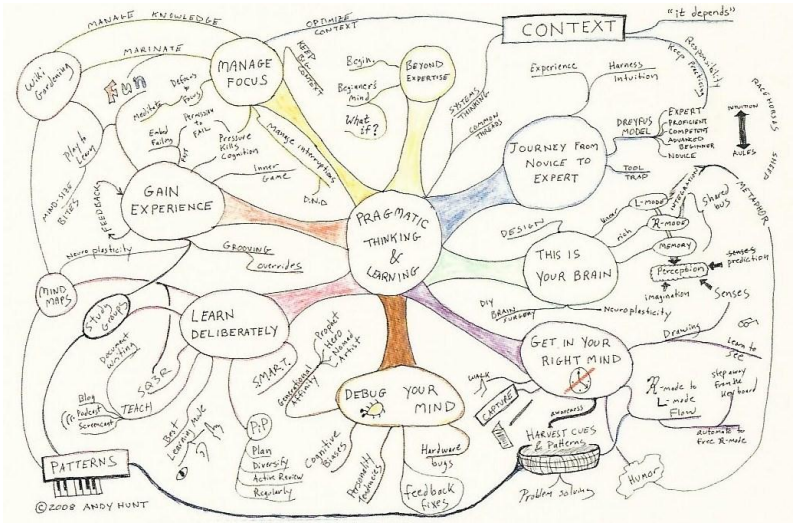


Figure 3-6: ***An Example of a Mind Map*** **[43].**

The use of mind maps in this study would allow access to the student's mental image of CS and their understanding of CS. In creating a mind map on CS, students would create a diagram with the topics they find relative to CS and how they interconnect. The mind map was to start with Computer Science as the root and students were to add as many nodes and subnodes as they see fit (see Figure 3-7); keeping in mind their connections to the root that was CS. Such a diagram would show how students view CS and provide a way of analyzing their understanding of it.

In this study, students were asked to submit two mind maps, one at the beginning of the semester and one near the end. Analyzing the change from the first mind map to the second would show the change in the students' understanding of CS from the beginning of the semester to the end. This would help investigate whether the course had an effect on the students' view of the CS and their understanding of it.

To analyze such diagrams, a count was implemented of the nodes, edges, and levels in each mind map. The levels in a mind map were counted by treating the map as a tree graph with the CS node as the root and ignoring any cycles; here the depth of the tree is the number of levels. For example, a student may have connected three nodes to the root node (CS) and to each of those he connected 2 nodes with some nodes interconnecting and some outliers (see Figure 3-7). A count would result in nine nodes, eleven edges, and three levels for that mind map. The resulting numbers were then tested for any statistically significant differences.

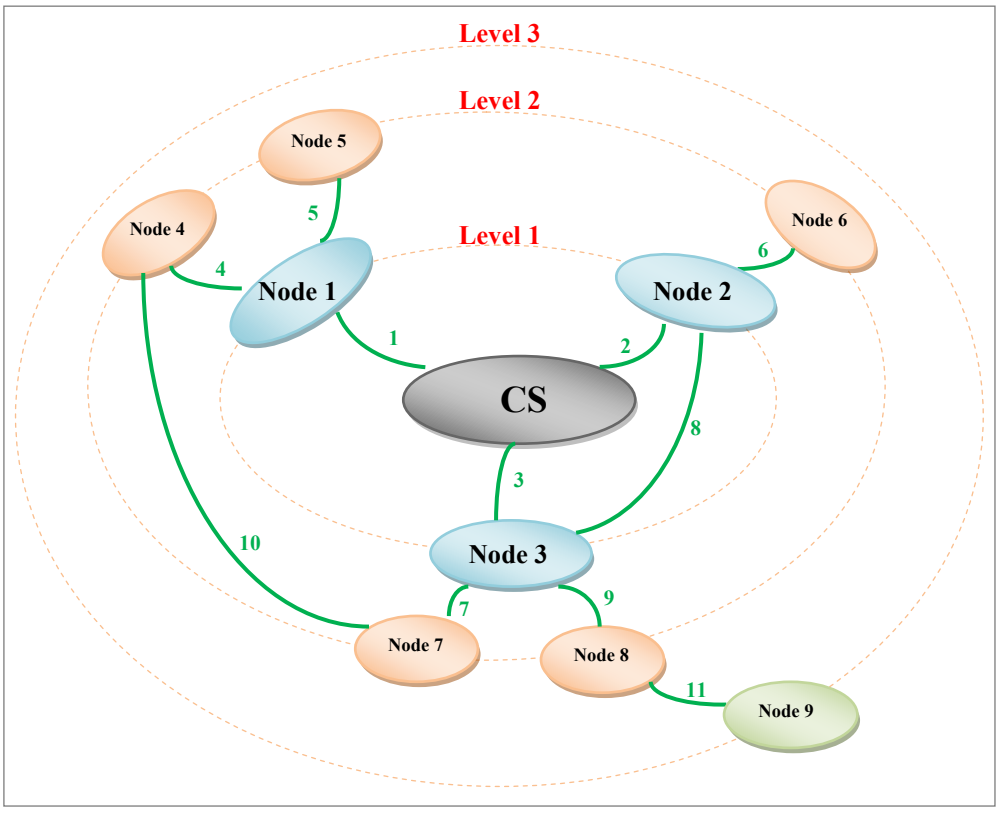


Figure 3-7: ***An Example of Counting Elements in a Mind Map.***

To simply count the number of nodes, edges, and levels in a mind map was not enough. This type of analysis can be too narrow from a cognitive perspective. Some of the nodes or subnodes included topics not relevant to CS. Therefore, to examine whether those connections and nodes were of any significance, that they were not just randomly added words of students seeking simply to mask their indifference, a closer inspection of the two mid maps was crucial. There are several coding systems for evaluating mind maps [e.g. 65,67, 64]; some depend on the use of a master map created by an expert [e.g. 77]. However, these systems evaluate student learning and their integration of knowledge not their view, which was the aim of this analysis. Since in this study we are examining CS as a discipline, the examination of the mind maps should be based on a well devised and formal description of the discipline. Fortunately such a description is found in the *Computing Curricula 2001: Computer Science* [2] and its revision report titled the *Computer Science Curriculum 2008: An Interim Revision of CS 2001*[3]. In those reports, CS as a "body of Knowledge" (CS BoK) is described and organized into a three level hierarchy; starting with "area" as the highest level then "unit" and ending with "topic." This organization provided a good basis for examining the mind maps which included nodes that can be corresponded to the areas and units in the reports' CS BoK. Thus an elimination process was implemented where the nodes and subnodes that do not correspond to the areas and units in the CS BoK were removed from consideration. Many studies have shown the importance of going beyond the individual words within the nodes in evaluating mind maps [e.g. 47, 45, 46, 58, 59]; they suggest the use of propositions. There were subtle differences in their definition of propositions. They all agree, however, that a proposition is the meaning of a node relative to its connecting

node(s). McClure, Sonak, and Suen [59] found the use of propositions in assessing mind maps the highest in terms of reliability. Such propositions were found to be important in this CS BoK examination process. For example, a mind map may contain two nodes containing the term ‘design’; one is connected to a node containing ‘Hardware’ and the other is connected to a node containing ‘Graphics’. In this case these two ‘design’ nodes propose two different meanings, thus both must count. This examination process would result in a new node count for both the first and the second mind map and give a more accurate representation of the students view in relevance to CS.

At the end of the semester each student, moreover, was asked to write an essay comparing the two mind maps. This comparative essay was to address both the contents and form of the two maps. In other words, the comparison would be both quantitative (e.g. number of concepts and their connections) and qualitative (e.g. what the map reflects about your learning). The two mind maps and the comparative essay were analyzed and studied to infer any affect the study had on the students understanding of CS and its concepts. In order to accomplish that, the nodes, edges, and levels of each mind map were counted and the results were compared and analyzed.

3.5 Validity and Reliability

To ensure the study’s validity and reliability, a strategy was used known as *triangulation* [61]. Two types of triangulation were used in this study: the use of multiple methods and multiple sources of data. In using multiple methods, this study employed the use of interviews, reflection papers, mind maps, and the MSLQ to compare data from these different methods of data collection. In using multiple sources of data, this study

compared data at different times from each data collection method. For example, the participants were interviewed at two different times then the data collected was compared.

4. RESULTS

Participants in CS116 (*N*=18) attended a semester long introductory CS course that implemented a new experimental approach introducing AIA and following the teaching methodology of SBL. To study this experimental approach's effect on the students' motivation, achievement, and attitude towards CS, participants were evaluated using modified MSLQs and interviews. The MSLQ responses were analyzed using the Statistical Package for the Social Sciences (SPSS). Their grades, responses, and submitted course work throughout the semester were also analyzed. This study followed a triangulation methodology of analysis where interviews, MSLQ, and students' course work were studied and analyzed to identify the significance of any relationship under study. Instructors' observations throughout the course are included under related section as found appropriate. In CS110, participants (*N*=12) attended a traditional introductory CS course that followed the lecture based approach with lab sessions. That course followed its usual teaching methodology that was used in previous semesters.

The two groups, CS116 and CS110, were also statistically analyzed and compared using the MSLQ. The results are organized based on the various variables of the study: first, the MSLQs are analyzed and any relationships were studied. Second, the interviews

are explored and studied. Finally, the submitted course work of students in CS116 is examined including programming assignments, reflection papers, and created mind maps.

4.1 MSLQ

Participants (*N*=30) answered modified MSLQs each containing 68 items. For each item, participants indicated how true the statement in the item was of them on a scale from 1 to 7 (1= "not at all true," 7= "very true"). In that scale, according to the MSLQ manual [68], the scores from 4 to 7 are considered high scores and the scores from 1 to 3 are considered low scores. For each participant, the 12 scales were calculated "by taking the mean of the items that make up that scale" (p. 5) [68]. The results are organized based on the various variables of the study: first, relationship among the three MSLQs completed by students in CS116 - the experimental group; second, the relationship between the two MSLQs completed by students in CS110 - the traditional group; and third between those competed in CS116 and their corresponding questionnaires completed in CS110. In each of these categories, the data collected from the background section were also analyzed allowing the study of the impact of participants' study habits and attitudes towards science subjects, including CS, on their experience in the course.

4.1.1 CS116

Participants in CS116 (*N*=18), the experimental group, answered three modified MSLQs, an entry-MSLQ, a mid-MSLQ, and an exit-MSLQ. For each MSLQ, the responses were studied and the rated scales were calculated. As indicated in Table 4-1, all

participants scored in the 4-7 range, which the manual defines as high, on all scales in all three MSLQs with the exception of Peer Learning scale in the mid-MSLQ.

A dependent t-test was implemented to identify statistically significant differences, if any, between the results of the three MSLQs. Out of the participants in CS116, 16 students completed the mid-MSLQ and exit-MSLQ and only one student completed all three MSLQs; therefore, the dependent t-test could not be implemented to compare the entry-MSLQ with the others since only one student completed that questionnaire. Between the 12 scales of the mid-MSLQ and the exit-MSLQ the results of the dependent t-tests showed no statistically significant differences, $p>.10$ (see Table 4-2).

Background information was also elicited from the participants as part of the MSLQ. Such information included their study habits, their attitudes about this course, and whether it would affect any future decision they may take involving computer science. The specific questions and their results are listed in Table 4-3. Overall, the results showed that the participants in CS116 did not find the course to be difficult, did not feel intimidated by the course, and they were comfortable with collaborating with others. One interesting finding was that in the exit-MSLQ, participants indicated that they were more comfortable collaborating with other students than they are asking questions (M=4.25 and M=3.88 respectively): $t(15)=-2.087$, $p<.10$. In other words, participants generally were more comfortable asking a peer questions than asking an instructor.

Scale	Entry-MSLQ (*n*=1)	Mid-MSLQ (*n*=17)			Exit-MSLQ (*n*=17)		
	Mean	**Median**	**Mean**	**SD**	**Median**	**Mean**	**SD**
Intrinsic Motivation	5.5	5.6	5.3	1.202	5.8	5.6	0.753
Extrinsic Motivation	6.5	5	5.1	1.119	5.1	5.2	1.295
Task Value	6.3	6.2	5.4	1.482	6.1	5.3	1.040
Control of Learning Beliefs	5.3	5.5	5.4	1.142	5.5	5.6	0.953
Self-Efficacy for Learning and Performance	4.9	5.9	5.6	0.984	6	5.8	0.845
Elaboration	5.3	4.8	4.4	1.238	5	4.9	1.214
Critical Thinking	4.8	4.6	4.6	0.735	5.1	4.8	0.746
Metacognitive Self-Regulation	4.3	4.6	4.5	0.968	4.8	4.7	1.030
Time and Study Environment	6	5.3	5.2	1.988	5.4	5.3	1.056
Effort Regulation	6	6	5.6	0.884	5.8	5.8	0.922
Peer Learning	4.3	4	3.8	1.300	4	4.1	1.481
Help Seeking	4.5	4.9	4.82	0.874	5.3	5	1.043

Note. Possible responses range from 1 (not at all true for me) to 7 (very true for me).

Note. Median values were added, in addition to the mean values, to provide better representation of the statistical data due to the small number of participants.

Table 4-1: ***MSLQ Scales' Means and Medians for CS116.***

Scale	Dependent t-test	
	t(15)	sig
Intrinsic Motivation	-1.163	.263
Extrinsic Motivation	0.379	.710
Task Value	-0.818	.426
Control of Learning Beliefs	-0.626	.541
Self-Efficacy for Learning and Performance	-0.382	-708
Elaboration	-0.867	.399
Critical Thinking	-0.951	.357
Metacognitive Self-Regulation	-1.027	.321
Time and Study Environment	0.761	.458
Effort Regulation	0.198	.846
Peer Learning	-0.569	.578
Help Seeking	-1.124	.279

Table 4-2: ***Dependent T-Test Between Mid and Exit-MSLQs for CS116.***

Question	Mid-MSLQ (*n*=17)			Exit-MSLQ (*n*=17)		
	Median	**Mean**	**SD**	**Median**	**Mean**	**SD**
How many hours per week do you prepare for this course? (1=0, 2=1-5, 3=6-10, 4=11-20, 5=21+)	2	2.44	.511	2.5	2.56	.629
How difficult do you find this course? (1=not difficult at all, 5=very difficult)	2	2.22	.548	3	2.69	.873
How likely is it that you will take another computer science course? (1=very unlikely, 5=very likely)	5	4.28	.958	5	4.44	.964
How likely is it that you will recommend this course to other students? (1=very unlikely, 5=very likely)	4	4.22	.808	4	4	1.155
How intimidated do you feel in the class? (1= very intimidated, 5=not at all intimidated)	3.50	3.50	1.150	4	3.81	.911
How comfortable are you in the following: (1= not at all comfortable, 5=very comfortable)						
asking questions	4	3.94	1.162	4	3.88	.806
collaborating with other students	4	4.11	.963	4	4.25	.577

Table 4-3: ***Background Information from CS116.***

Another dependent t-test was implemented; this time, however, testing if there was any statistically significant differences in the answers elicited from the background information section in the mid and the exit-MSLQ administered in CS116. As indicated in Table 4-4, only two statistically significant differences were found. First, there was a slight decrease in recommending the course to other students: $t(15)= 2.087$, $p<.10$. Nevertheless, in both the mid-MSLQ and the exit-MSLQ, participants indicated that it was 'likely' that they would recommend the course to other students (M=4.22 and M=4 respectively). The other statistically significant difference found was regarding the difficulty of the course: $t(15)= -1.464$, $p<.10$. Participants found the course more difficult when asked in the exit-MSLQ than in the mid-MSLQ (M=2.69 and M=2.22 respectively). Similarly, in both mid- and exit-MSLQ participants did not find the course difficult.

Question	Dependent t-test	
	t(15)	Sig
How many hours per week do you prepare for this course?	-0.293	.774
How difficult do you find this course?	-1.464†	.164
How likely is it that you will take another? computer science course?	-1.000	.333
How likely is it that you will recommend this course to other students?	2.087*	.054
How intimidated do you feel in the class?	-0.813	.429
How comfortable are you in the following:		
asking questions	0.368	.718
collaborating with other students	-0.808	.432

* $p<.10$, two-tailed. † $p<.10$, one-tailed.

Note. A one-tailed test was implemented since this study hypothesized a positive effect of the study factors (directional).

Table 4-4: ***Dependent T-Test on Background Information from CS116.***

4.1.2 CS110

In CS110, the traditional group, participants (*N*=12) answered two modified MSLQs each containing 68 items, an entry-MSLQ and an exit-MSLQ. For each item, participants indicated how true the statement was of them on a scale from 1 to 7. For each modified MSLQ, scale scores were calculated (see Table 4-5). With the exception of Peer Learning and Help Seeking, participants scored on the high range, between 4 and 7, on the scales in both MSLQs.

Out of the 12 participants in CS110, only one student completed both entry and exit-MSLQs, five completed only the entry-MSLQ, and six completed the exit-MSLQ. Therefore, instead of a dependent t-test, an independent t-test was implemented on this group (see Table 4-6). Similar to CS116, no statistically significant differences were found, $p>.10$.

As part of the MSLQ, background information was elicited from the participants. Tables 4-7 and 4-8 below list the means and medians for the answers to the questions from the background information section in the mid and exit questionnaires. In the entry-MSLQ, participants did not anticipate the course to be difficult, and from a list of factors that may have affected their decision to enroll in CS110 they chose 'will be useful to me in school', 'will be useful to me in life', 'will help improve my academic skills', 'will improve career prospects', and 'fit into my schedule' as important factors and 'was recommended by a friend' as not an important one (see Table 4-7.a and 4-7.b). On all of the other items included in this section of the questionnaire, they were neutral. In the exit-

MSLQ, participants were neutral on all the items listed in the background information section (see Table 4-8).

Scale	Entry-MSLQ (*n*=6)			Exit-MSLQ (*n*=7)		
	Median	**Mean**	**SD**	**Median**	**Mean**	**SD**
Intrinsic Motivation	5.1	5	0.697	4.5	4.9	1.412
Extrinsic Motivation	5.8	5.5	1.030	5.3	5.4	0.852
Task Value	5.6	5.3	1.040	5.8	5.2	1.340
Control of Learning Beliefs	5.6	5.4	0.848	4.8	4.6	1.079
Self-Efficacy for Learning and Performance	5.8	5.7	0.522	6	5.2	2.014
Elaboration	4.3	4.5	0.678	4.5	4.5	1.166
Critical Thinking	5	4.8	0.669	4.8	4.5	1.118
Metacognitive Self-Regulation	4.3	4.3	0.795	3.8	4.3	1.105
Time and Study Environment	4.8	4.7	0.498	4.6	5	0.924
Effort Regulation	5.4	5.2	0.765	6	5.6	1.125
Peer Learning	3.5	3.3	1.075	3	3.3	1.380
Help Seeking	3.4	3.5	0.592	3.8	3.8	1.475

Note. Possible responses range from 1 (not at all true for me) to 7 (very true for me).

Note. Median values were added, in addition to the mean values, to provide better representation of the statistical data due to the small number of participants.

Table 4-5: ***MSLQ Scales' Means and Medians for CS110.***

Scale	Independent t-test	
	t(5)	sig
Intrinsic Motivation	0.178	.862
Extrinsic Motivation	0.194	.850
Task Value	0.100	.922
Control of Learning Beliefs	1.342	.207
Self-Efficacy for Learning and Performance	0.574	.578
Elaboration	0.037	.971
Critical Thinking	0.655	.526
Metacognitive Self-Regulation	-0.084	.935
Time and Study Environment	-0.865	.405
Effort Regulation	-0.667	.518
Peer Learning	0.068	.947
Help Seeking	-0.443	.667

Table 4-6: ***Independent T-Test between Entry and Exit MSLQs for CS110.***

Question	Entry-MSLQ (*n*=6)		
	Median	Mean	SD
How many hours per week do you prepare for this course? (1=0, 2=1-5, 3=6-10, 4=11-20, 5=21+)	2	2.33	.516
How difficult do you anticipate this course to be? (1=not difficult at all, 5=very difficult)	2	2.00	.632

Table 4-7.a: ***Background Information from the Entry-MSLQ from CS110.***

Question	Entry-MSLQ		
	Median	**Mean**	**SD**
How likely is it that you will take another CS course? (1=very unlikely, 5=very likely)	4	3.83	1.169
How important were the following for your decision to take this course/class: (1 = not at all important, 5 = very important)			
fulfills a course requirement	4.5	3.67	1.751
experience seemed interesting	4	3.50	.837
is required	5	3.67	2.066
will be useful to me in school	4	4.00	1.095
will be useful to me in life	4	4.33	.516
will help improve my academic skills	4	4.17	.753
was recommended by a friend	1	2.00	1.673
was recommended by an advisor/professor	3.5	3.33	1.633
will improve career prospects	4	4.00	.894
fit into my schedule	4	4.00	.894
How confident are you in the following: (1 = not at all confident, 5 = very confident)			
Engineering	3.5	3.00	1.265
Math	4	3.67	.516
Reading	4	4.33	.516
Writing	4	4.17	.753
Science	3	3.33	.516
computer science	3	3.33	1.033
How comfortable are you in the following: (1= not at all comfortable, 5=very comfortable)			
asking questions	4	3.50	.837
collaborating with other students	4	3.83	.408

Table 4-7.b: ***Background Information from the Entry-MSLQ from CS110.***

Question	Exit-MSLQ (*n*=7)		
	Median	**Mean**	**SD**
How many hours per week do you prepare for this course? (1=0, 2=1-5, 3=6-10, 4=11-20, 5=21+)	2	2.14	.378
How difficult do you find this course? (1=not difficult at all, 5=very difficult)	2	2.71	1.380
How likely is it that you will take another computer science course? (1=very unlikely, 5=very likely)	4	3.71	.951
How likely is it that you will recommend this course to other students? (1=very unlikely, 5=very likely)	3	3.00	1.000
How intimidated do you feel in the class? (1= very intimidated, 5=not at all intimidated)	4	3.14	1.574
How comfortable are you in the following: (1= not at all comfortable, 5=very comfortable)			
asking questions	3	3.43	.976
collaborating with other students	4	3.71	.756

Table 4-8: ***Background Information from the Exit-MSLQ from CS110.***

The data collected from the background section of the questionnaire was tested for any statistical significance between the entry-MSLQ and the exit-MSLQ. Only four items are similar between the entry- MSLQ and the exit-MSLQ. They are listed in Table 4-9. An independent t-test was implemented on these items and none were found to be statistically significant, $p>.10$.

Question	Independent t-test	
	t(11)	Sig
How difficult do you anticipate/find this course to be?	-1.162	.270
How likely is it that you will take another computer science course?	0.203	.843
How comfortable are you in the following:		
asking questions	0.140	.891
collaborating with other students	0.344	.738

Table 4-9: ***Independent T-Test on Background Information from both MSLQ's from CS110.***

4.1.3 A Comparison Between CS116 and CS110

To analyze the differences in the 12 scales between CS116 and CS110, an independent t-test was performed on the exit-MSLQ. The entry-MSLQ was not tested due to the fact that only one student in CS116 completed the entry-MSLQ. The results for

the 12 scales in the exit-MSLQ are listed in Table 4-10 and illustrated in Figure 4-1 below.

Scale	Independent t-test	
	t(21)	sig
Intrinsic Motivation	1.522†	.143
Extrinsic Motivation	-0.287	.777
Task Value	1.014	.322
Control of Learning Beliefs	2.119**	.046
Self-Efficacy for Learning and Performance	1.136	.269
Elaboration	0.796	.435
Critical Thinking	0.776	.446
Metacognitive Self-Regulation	0.880	.389
Time and Study Environment	0.459	.651
Effort Regulation	0.400	.693
Peer Learning	1.307	.205
Help Seeking	2.295**	.032

** $p<.05$, two-tailed. † $p<.10$, one-tailed.

Note. A one-tailed test was implemented since this study hypothesized a positive effect of the study factors (directional).

Table 4-10: ***Independent T-Test for Exit-MSLQs in both CS116 and CS110.***

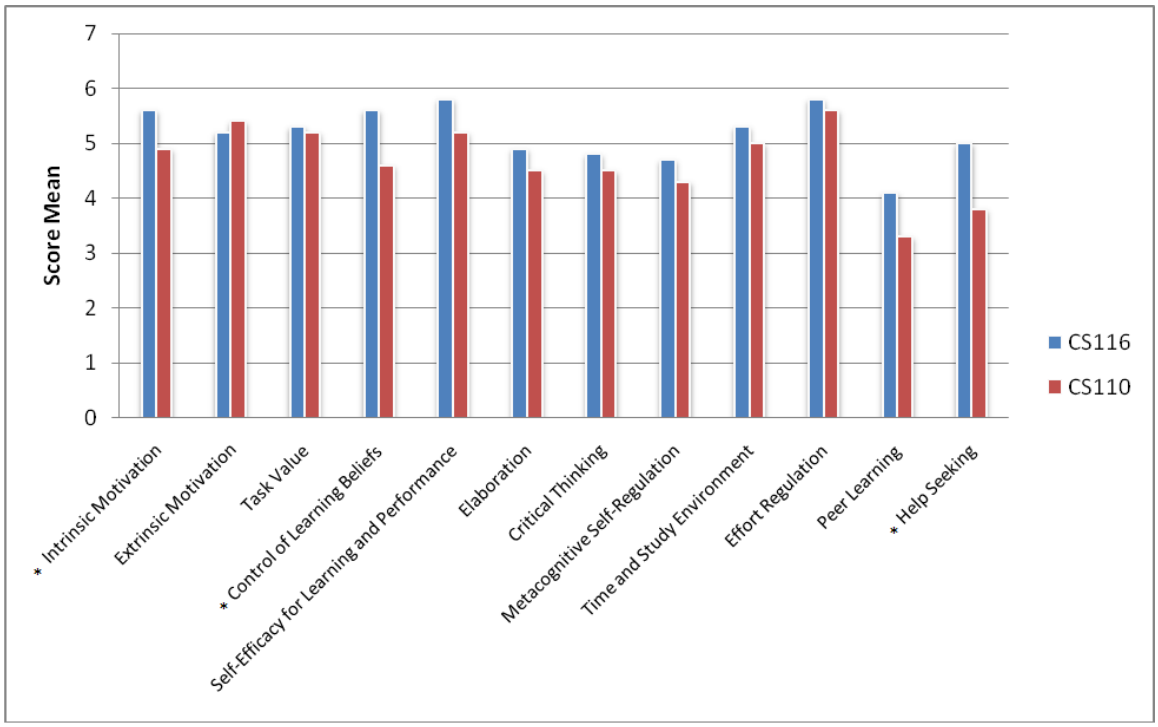


Figure 4-1: ***CS116 and CS110 Exit-MSLQ Chart.***

For the exit-MSLQ, Intrinsic Motivation, Control of Learning Beliefs, and Help Seeking scales were found statistically significant ($t(5)= 1.522$, $t(5)= 2.119$, and $t(5)= 2.295$ respectively). Participants in CS116 rated higher in those three scales than participants in CS110. It is however worth mentioning that, with the exception of Help Seeking, participants in CS110 scored on the high spectrum as well.

The differences between the background information in the exit-MSLQ collected from CS116 and CS110 were also analyzed for statistical significance. To do so, yet another independent t-test was administered (see Table 4-11). Statistically significant differences were found in the likelihood of recommending the course to other students and in the likelihood of taking another CS course. In the former, participants in CS116

were more likely to recommend their course, the experimental course that is, than participants attending CS110 recommending theirs, the traditional course (M=4.00 and M=3.00 respectively): $t(21)=1.983$, $p<.10$. Nonetheless, participants in CS110 were neutral in the likelihood of recommending their course. In the later, participants in CS116 rated higher in the likelihood of taking another CS course than participants in CS110 (M=4.44 and M=3.71 respectively): $t(21)=1.662$, $p<.10$. One more item was found to have a statistically significant difference, the comfort level in collaborating with other students. CS116 participants were more comfortable collaborating with other students than CS110 participants (M=4.25 and M=3.71 respectively): $t(21)=1.866$, $p<.10$.

Question	CS116	CS110	Independent t-test	
	Mean	Mean	t(21)	sig
How difficult do you find this course? (1=not difficult at all, 5=very difficult)	2.69	2.71	-0.057	.955
How likely is it that you will take another computer science course? (1=very unlikely, 5=very likely)	4.44	3.71	1.662†	.111
How likely is it that you will recommend this course to other students? (1=very unlikely, 5=very likely)	4.00	3.00	1.983*	.061
How intimidated do you feel in the class? (1= very intimidated, 5=not at all intimidated)	3.81	3.14	1.296	.209
How comfortable are you in the following: (1= not at all comfortable, 5=very comfortable)				
asking questions	3.88	3.43	1.148	.264
collaborating with other students	4.25	3.71	1.866*	.076

* $p<.10$, two-tailed. † $p<.10$, one-tailed.

Note. A one-tailed test was implemented since this study hypothesized a positive effect of the study factors (directional).

Table 4-11*: Independent T-Test on Background Information from the Exit-MSLQ between CS116 and CS110.*

4.2 Interviews

Participants from CS116 volunteered to answer questions in two interviews, an entry interview and an exit interview. Both interviews were voluntary and anonymous. These two interviews aimed at gaining an insight into the participant's attitudes towards this CS course in particular as well as towards CS as a major. The point was to examine whether this experimental course had an effect on the students' attitudes, what type of effect it was, and what in particular might have been the factors. A total of eight students participated in the interviews, one in the entry and seven in the exit.

4.2.1 Entry Interview

One student completed the entry interview and answered a total of eleven pre-set questions (see appendix K for transcripts). The interview took place in a private room at the university's library. It was administered by a graduate student who volunteered for the job.

After formal greetings, the interviewer started with asking the interviewee about the reason why he enrolled in the course. The interviewee responded that the course completed the elective requirements of his major but failed to specify his exact major. The interviewee has used computers before but had no programming experience, which was expected since this is a CS course for non-CS majors.

The interviewee never considered CS as a major. He stated that when it comes to computers he would prefer dealing with hardware rather than software. The interviewee was asked a few questions to gain a perspective on his mental image of who a computer scientist is. His view was very simplistic and showed the common attitudes often

exhibited towards the discipline. He defined them as people who study and research computers. Even though he acknowledged the requirement of specific skills in a computer scientist, he did not know what they were. The only skill he mentioned was "patience;" as he puts it, if computers "are going slow so you know to not get mad about it." All of the above showed a negative attitude and a preconceived frustration towards the study of computer science and the typical mentality that CS is synonymous with programming.

The interviewee, moreover, was asked questions to evaluate his comfort level in collaborating with others. He responded very strongly in favor of solo work and mentioned, as he puts it, "its advantages," such as being able to concentrate more. He was further asked what he thought about peer evaluation. At first the interviewee did not know what peer evaluation was, after some clarifications he stated that he found such type of evaluations misleading. He preferred instructor evaluations "because they know what they are grading on," with that statement he showed a focus on the end grade rather than on the learning journey. With that the interview was concluded and the interviewee was thanked for his time.

An emphasis is needed on the fact that only one student volunteered for the entry interview. His perspective is not representative of the students attending CS116. The aim of this interview was to provide comparative data to those collected in the exit interview; this can no longer be implemented. Nonetheless, this interview did provide valuable qualitative data about that student's attitude and perceptions.

4.2.2 Exit Interview

With the exit interview, responses were elicited by email due to the very low number of participants in the entry interview. An online form was created and the students were sent an email, with a link to that form, and asked to participate and answer some questions. Similar to the entry interview, the aim was to extract the students' attitudes towards the course in particular and towards CS in general. Seven students responded and filled out the form online (see appendix L for the complete responses).

Background. Respondents were first asked a couple of questions to confirm that they had little to none programming experience previous to this course. Out of the seven respondents, only one student took a CS course before and had some experience with HTML. Again this was an expected, even a wanted, result since the course was open to non-CS majors.

AIA. As the literature predicted, creating mobile applications did spark the respondents' interest. The respondents were excited about that fact that they were able to create working applications while using AIA as a visual blocks programming language. The instant gratification they got from creating the mobile applications increased their enthusiasm in class. One respondent commented on that in this interview, "my favorite aspect of using App Inventor was that the applications you create actually work for the phones and that is really cool to know that you created this app." "I liked the fact that we learned the basics of what makes applications work and how to design our own," another respondent elaborated. The drag and drop of visual blocks was found to be helpful in the programming process. The students were able to focus on the structure of a program

block and understand it. "It did make the thought process way easier," one respondent elaborated. They did not have the struggles and distractions often associated with textual syntax. Overall, AIA was found to be very easy to work with and was described as "user friendly." In fact they found that AIA provided a space for culturing creativity,

> The projects were a lot of fun. I love being creative whenever I can so this was another awesome outlet for that creativity . . . [and] now I can create so many more useful things, even make things for use in other courses. (Respondent)

Respondents however did divulge some struggles. One respondent exemplified that in his statement: "I loved it, when it cooperated." Respondents found AIA was "very complicated at times" especially in figuring out which visual blocks to choose when building an application. They got a bit frustrated about bugs that would appear for "no apparent" reason and "the amount of time it took to get something to work." The AIA IDE did freeze at times and non-significant error messages did pop up from time to time. This was expected since AIA was an experimental IDE and was still under construction at the time of this experimental course. Over the semester, several changes occurred to the IDE which one respondent found a bit frustrating, "[it] messed with some of our projects." It might be worth mentioning here, however, that the instructors noticed that debugging was less of an issue in this course compared to other CS courses. Their observations suggested that the majority of class time was spent on design issues and understanding of CS concepts.

SBL. A series of questions were also included to examine the respondents' attitudes toward components of SBL, mainly collaboration and *design crits*. All respondents found SBL beneficial, some even found it enjoyable,

> They allowed us to meet people and expand my learning. I was more willing to reach out to people for help. . . . I thought sometimes it was challenging. . . . I think that the projects went better than I expected it would go on the first day. (Respondent)
>
> I learned a lot . . . the projects were very fun and I liked that the students were in full control over the projects that they were working on. (Respondent)
>
> I really liked how we were able to work in groups to do our projects and the groups changed a few times so we could work with new people and get to know them better. (Respondent)
>
> I love studio based learning and I learn more in these kinds of environments. (Respondent)

One respondent did not share those sentiments, however he did not seem interested in CS courses at all, "this class taught me to think more logically but that would be about it, since I dont[*sic*] plan to do any programming in the future."

Respondents had mixed reactions when it came to collaboration; most found it very comfortable, a few did not, and some found it comfortable enough "to get the task done." One respondent liked it so much that he requested more collaborative work, "I really like the way this course is design. I would like to see more group projects instead of individual projects." To one respondent familiarity was important, "I do not like to

work with other people unless I've known them a long time and know how to deal with them." Familiarity here provided the essential comfort level for this respondent to perform in a collaborative environment. To another respondent this experience was a turning point, "this class helped me to become more open to working with others[*sic*] classmates to solve the problems that would arise."

Instructor vs. peer evaluation. Additionally, the respondents' preference for evaluation type, whether peer or instructor, was examined. Almost half of the respondents did not have a preference, seeing both types of evaluation as valuable. The other half of the respondents preferred instructor evaluation stating that "peer evaluation has a tendency to be less honest." They perceived peer evaluation to be "less reliable" and that "peers didnt[*sic*] care as much to give constructive feedback." One respondent, however, noted how knowing that peers will be evaluating the performance gave students motivation to perform better; commenting that peer evaluations tend to be "more of an encourager than a giver of positive feedback." Overall, most of their comments were along the line of how peer evaluation is not reliable not how instructor evaluation is better.

The course en bloc. In the exit interview, all respondents felt the course had met their expectations; which were to learn programming basics and be able to actually apply what they learned. In fact, a few of them stated it exceed their expectations. "It went above and beyond my expectations. This is the first time at Ball State that I have ever had the so called 'immersive learning' experience," as one respondent explained. For BSU students this statement carries great significance. It is a phenomenon that took over the university and became its mission statement. At BSU, "Immersive learning" signified

creating a real-world hands-on problem solving environment that would prepare the students for their future careers and give them the ability to compete in the industry. It is however important to clarify that this course was not built with this mission statement in mind. Furthermore, according to the university's definition this experimental course was not an immersive learning experience; it did not involve an off campus experience with a community partner nor did it produce a final product that impacted the community. Nonetheless, the respondents had a positive learning experience in this experimental course,

> I learned a lot because I entered this class to[*sic*] little to none programming experience and my skills with the programming grew a great deal over the course of this semester. (Respondent)
>
> I learned a lot about programming [and] how different aspects work cohesively . . . [and] the projects were all worth the effort and everything made you get outside your comfort zone. (Respondent)
>
> I really wouldn't change anything. I love the ability to freely develop and do whatever comes to my mind. Restricting people only holds the mind back and students just don't need any more of that. (Respondent)
>
> I would not really change anything about the course. I thought it was perfect the way it was. (Respondent)

One respondent did not agree with his fellow respondents. His comments, however, related to the use of formative assessment in the grading process. He felt that "the students should have more say in the grade they receive rather than just being assigned a letter grade based on what the instructors have observed." It is essential, however, to point out that after each grading process the instructors gave the students a chance to discuss their grade if they felt any injustice and were willing to make changes accordingly.

Attitudes towards CS. Now that the experimental course was over, participants in this exit interview were asked questions eliciting their attitudes towards CS. Almost all respondents felt they experienced a change in their view of CS. Before this course, a respondent felt that CS was rather difficult and not anybody would be able to understand it; a view that later changed, "I looked at computer science as something only nerds could do but it is actually rather simple." As another respondent explained his view of CS he showed a broadened view of CS and an understanding that it's not just about computers and programming, "computer science is important in the fact that it is involved in everything." "I began thinking that Computer Science wasn't a very big deal but in the end I realized that Computer Science deals with many different aspects of the world that we live in today," another respondent stated. Respondents recognized that computer science went beyond programming to involve problem solving, critical thinking, and "high understanding of mathematics." The anti-social image CS has as a discipline experienced a change to a more social one as well. Respondents listed collaboration as a skill a computer scientist should have. This was due to the social learning environment

SBL provides. This changed view resulted in a positive attitude toward CS. Respondents felt a deeper appreciation of the discipline and an increased interest in CS,

> Coming into college, I wasn't exactly sure what[‘s] the difference between a CS major and a IT major. This made me appreciate the CS major more and I definitely hope to take more classes on this material. (Respondent)
>
> It gave me a better insight into the development of programs and applications which is something I didn't truly know before. (Respondent)

The one respondent that replied negatively to the change in view towards CS stated: “I have the same view as before.” To understand his ‘before’ view of CS, a look at his other responses was needed. This respondent was one of those whom had no programming experience. However, he majored in Computer Technology. His expectancy of this course was to get “an introduction to the basic concepts of code . . . through the use of mobile applications” which was the main goal of this course. When asked what skills a computer scientist should have, he stated “coding skills” and “basic PC knowledge.” Though he did not elaborate on what would be included under ‘coding skill’, he did show an understanding that CS is not all about the computer. That respondent stated: “I have always been interested in computer science since I began exploring into the computer world with the purchase of my new computer,” showing that he has had interest in CS prior to this course and may have collected some basic information related to CS along the way; curiosity often leads to questions which itself leads to research. Overall, this

respondent showed a slight understanding of what CS is thus did not feel he experienced change in his view.

Five of those respondents, however, still showed a misunderstanding of what CS as a discipline is really about. One respondent acquainted it with programming and only programming, “my personal life doesn't really involve in[*sic*] programming but it may later when I have a career.” Another stated that he only used CS “for entertainment,” probably thinking of the computer. “Every time I get on the internet or do anything with my computer I'm dealing with computer science,” yet another respondent stated. One respondent believed that “if you can know as much as possible with[*sic*] computers than[*sic*] computer science will come easy.” One respondent felt that all computer science did for him is further his “knowledge of computers.”

4.3 Mind Maps

Near the beginning of the semester, students were asked to create a mind map with CS at the root. Through this map, students illustrated their understanding of CS and any connections they deemed related to it before they start the course. Near the end of the semester, students were asked once again to create a new mind map with CS at the root. With this map, students illustrated any change in their understanding of what CS is and what is related to it. Sixteen participants submitted both mind maps, however two participants misunderstood the task on the second map thus their maps were removed from consideration in this study.

Fourteen participants’ mind maps were analyzed for any significant differences. First, a count was done on each map for the number of nodes, edges, and levels (see

Table 4-12). A dependent t-test was performed to test for any statistically significant differences between the first and second mind map (see Table 4-13). Statistically significant differences were found in all three elements. Participants more than doubled the number of nodes in their second map indicating that they found more categories to connect to CS from the first to the second mid map (M=15.67 and M=35.14 respectively): $t(13)= -3.499$, $p<.01$. The increase in number of edges from the first to the second was expected since they connect nodes which, as stated above, had increased (M=19.93 and M=39.79 respectively): $t(13)= -3.255$, $p<.01$. The participants also went a level deeper in their mind maps of CS indicating they found connections that went further than the obvious ones (M=3.07 and M=4.36 respectively): $t(13)=-3.628$,$p<.01$. The increase in these three elements showed a significant change in the participants' view from the beginning of the semester to the end.

Category	**Entry - Mid Map (*n*=14)**		**Exit - Mind Map (*n*=14)**	
	Mean	**SD**	**Mean**	**SD**
Number of nodes	15.67	5.205	35.14	23.178
Number of edges	19.93	6.341	39.79	24.473
Number of levels	3.07	.704	4.36	1.447

Table 4-12: ***Means for Elements of the Mind Maps.***

Elements	Dependent t-test	
	t(13)	Sig
Number of nodes	-3.499***	.004
Number of edges	-3.255***	.006
Number of levels	-3.628***	.003

*** $p<.01$

Table 4-13: ***Dependent T-Test for Elements of the Mind Maps.***

In order to examine whether the change in view was related to the participants' understanding of CS and the way they viewed CS, the submitted mind maps were examined using the CS BoK. The nodes in the mind maps that did not correspond to areas and units in the CS BOK were eliminated which resulted in a new node count (see Table 4-14). After this elimination process, a dependent t-test was administered and the difference between the number of nodes in first mind map and in the second was found to be statistically significant: $t(13)=-2.709$, $p<.02$. These results indicated that participants had a significantly positive change in their image and understanding of CS and were able to find more connections and topics relating to CS in the second mind map than in the first ($M=10.07$ and $M=5.93$ respectively).

Number of Nodes	Mean	SD
First mind map	5.93	1.774
Second mind map	10.07	6.639

Table 4-14: ***Mind Maps After Examination Using CS BOK.***

A qualitative analysis was implemented of the mind maps and the comparative essays written by the students. Specific observations were noted on each mind map and on the differences between the two. Participants showed, on different levels, an improvement in their understanding of CS as a discipline from the first mind map to the second. Even though programming was mentioned equally in both mind maps, participants were able to make many new connections and expand their image of CS to go beyond programming. This was a desirable outcome for programming is an integral element of CS but not the only element, and participants were able to see that. One participant explained that while creating his first mind map he "could not think of what to put on the mind map. The second time around it was like the exact opposite;" he "could not stop writing things down." That participant did make many new valid connections to CS in his second mind map and showed improvement in his understanding. He found that CS connects to other disciplines, such as biology and chemistry. Others participants found relations to other disciplines in their second mind map as well, such as engineering and art. A different participant also found new relations to CS and connections beyond

the discipline itself. She established connections to CS as a place for innovative ideas and "helping needs,"

> I have greatly expanded my knowledge of how people, computers, graphics, and applications are related. I have realized it takes more then[*sic*] just an idea it takes the knowledge of logic and how things will fit together to undergo procedures. . . . I also realized applications are not always just for entertainment, applications are also made to make things easier in life.

Moreover, through her experience in this course she found that "computer science can be fun and rewarding." She found creating applications using visual blocks programming motivational and rewarding even though her previous view about programming was that it's "boring." Another participant commented on his overall change of view in his essay, "this course has helped refine my idea of what Computer Science is;" enough so that he was "inspired" to change his "major back to computer science and continue work in the field." That participant did show an improvement in his mind maps, however when compared to other participants his maps showed minute change.

Some participants saw the need for people skills in CS, such as communication and collaboration; connections that did not exist in their first mind map. A couple of participants incorporated creativity in both mind maps. This indicated that those two participants came into the course with the idea that it would provide an outlet for creativity and in the end to them it did. Along those lines, one participant commented in his essay stating that the course had "been a great experience in the innovative world of application development." Another participant focused in her first map solely on

computers. She wrote in her essay about how her view changed to see how different devices, such as “Playstation” and “eReader,” are related to CS and required an understanding of CS concepts as well,

> For them to all feel so separate on the surface and all be so similar on the inside was an interesting thought. I knew these things all along but actually thinking about it I must attribute to the whole Android experience. . . . That says to me that computer science is a very unifying field.

On the other hand, one participant did not show much interest in CS in both mind maps and seemed more concerned about getting a good grade. Another participant’s mind maps did not show much progress either in his understanding of CS as a discipline. He went from finding CS exciting in his first mind map to being a “headache” and “full of surprises” in his second. Overall, however, the observations were consistent with the finding above; they showed that participants by the end of the semester had a better understanding of CS than at the beginning.

4.4 CS116 Student Course Work

Throughout the semester, the students in CS116 completed a series of programming assignments. Upon completion of which, they wrote reflections papers describing their experience. The student also created two mind maps, one at the beginning of the semester and one at the end, and wrote an essay analyzing the two. All of which, along with in class observations, are declared below.

4.4.1 Assessment

Students, for the most part, were doing very well throughout the semester. It was evident in their submitted work that they were making use of the feedback given, both formal and informal, in the process of formative assessment. Students were able to actively learn with very few interjections from the instructors. It is important to note, however, that at first students did not relate to the idea of formative assessment and were always seeking ways were they could get a tangible grade on paper. For the most part, however, students did appreciate the process of formative assessment and did find it very helpful in their learning process.

In the process of summative assessment, students were split between the two instructors for evaluation. Due to context sensitivity and the subjectivity of evaluators, no attempt was made to correlate the grades with any of the data collected. Below is simply a listing of students' grades and a discussion of their progress.

Students were assessed in the middle of the semester through assessment-by-interview (see section 3.1.1 above). Students were shown a sequence of blocks. Then they were asked to explain what the code was doing and what the expected result was. Overall, the students did well in this midterm assessment (see Table 4-15). Over half of the students were above the C letter grade with approximately 12% getting an A, 41% getting a B, 35% getting a C, and 12% getting a D. No student failed this assessment process. The concepts that some of the students struggled with in this assessment-by-interview included string operations–nesting of the 'join' command specifically–and the difference in the logic of 'if' vs. 'ifelse'. A few students seemed to have difficulty

understanding the difference between defining a variable and setting a value to a variable. Generally, there were some concerns about the ability of those struggling students to turn their grades around and do better at this stage.

Grade	Frequency	Percent (%)	Cumulative Percent (%)
A	2	11	11
A-	0	00.0	
B+	0	00.0	
B	7	39	50
B-	0	00.0	
C+	0	00.0	
C	7	39	89
C-	0	00.0	
D	2	11	100
F	0	00.0	

Table 4-15: ***Midterm Grades Frequency Distribution.***

At the end of the semester, the students submitted a four week long final project (see section 3.1.1 above). In that project, students were expected to show their ability to work with AIA and implement some of the complex concepts in CS covered throughout the semester. The majority of the students did very well. They had shown an understanding of the basic concepts covered in the course, some of them were concepts

they had struggled with in their midterm exam. The final grade was assigned taking into consideration the cumulative work of each student and the progress they have shown throughout the course. Overall students did very well where 47% of the students got an A, about 12% got an A-, 6% got a B+, approximately 23% got a B, and about 12% got a D (see Table 4-16). No students failed the course. One student, however, withdrew from the course. These results indicated that the experimental course had a high success rate with 83% of the participants getting a grade between A and C, that is, not withdrawing, failing, or getting a D.

Grade	**Frequency**	**Percent (%)**	**Cumulative Percent (%)**
A	8	47	47
A-	2	11.8	58.8
B+	1	6	64.7
B	4	23.4	88.2
B-	0	00.0	
C+	0	00.0	
C	0	00.0	
C-	0	00.0	
D	2	11.8	100
F	0	00.0	

Table 4-16: ***Final Grades Frequency Distribution.***

A comparison between the midterm grade and the students' final grade was implemented (see Figure 4-2). Due to the difference in measurement methods utilized in both assessment periods, this comparison was in terms of grades only, not student's achievement or comprehension. With that said, this individual comparison showed an improvement in terms of grades from the middle of the semester to the end. The majority of the students were doing better by the end of the semester, getting B and above with almost half getting an A. At the end of the semester the grades were emailed to the students a few days before finalization. The students were informed that they have the opportunity to discuss their grades and appeal for any changes. None of the students took this opportunity nor contacted either of the instructors for an appeal.

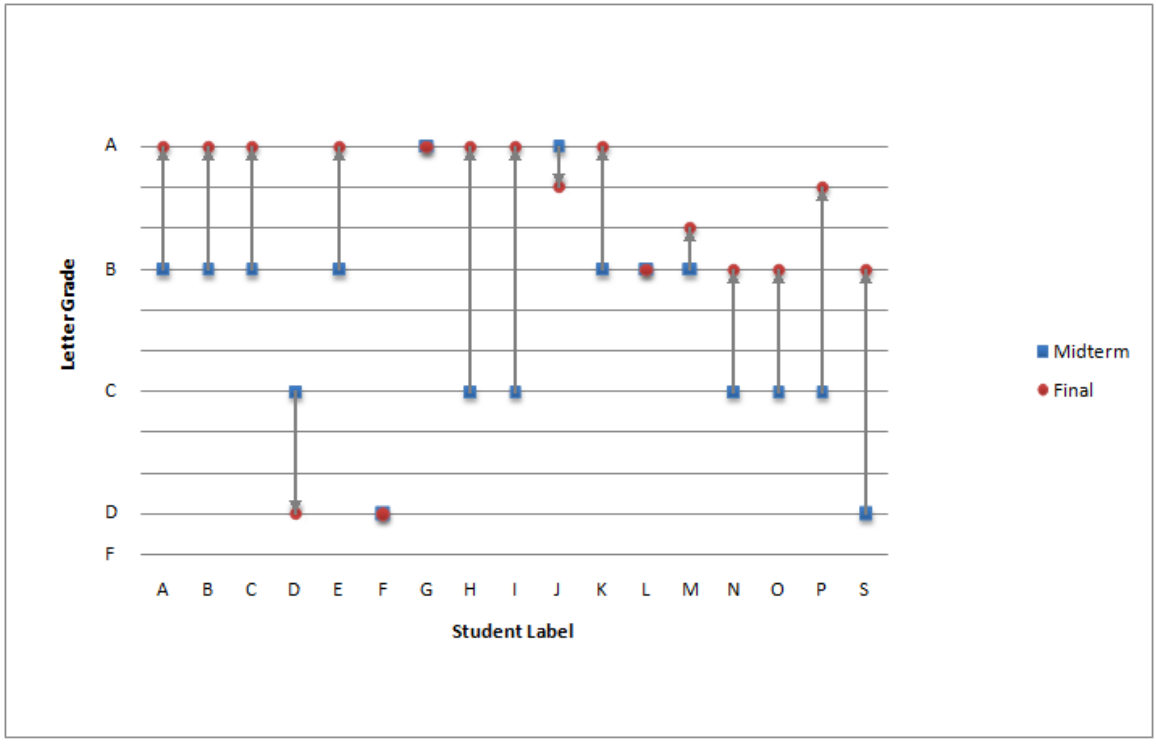


Figure 4-2: ***Individual Grade Progress Linechart.***

4.4.2 Programming Assignments

All the students in CS116 completed all programming assignments and created functioning applications that worked on the G1 phones. Keeping in mind that students had to find the solutions to the assignments on their own, this demonstrated the ease of use AIA provided as the visual blocks programming language. For some projects a theme was specified, for others students were given the freedom to choose the type of application they wanted to create. This section includes some of the projects that stood out throughout the semester. These projects contained unconventional elements that qualify them to be described as having p-creativity. They are listed here to show that students did exhibit such creativity in their work. More importantly, they show how with the right tools and learning environment such creativity can be fostered. A larger list of projects is available in Appendix M.

One theme for a project was Halloween, simply because it was close to Halloween. A student created an app that he called "PumkinHead." According to the student, it was an app inspired by "Mr potato head" where the user would change the face's eyes, nose, and ears shape. The student did not want to use buttons- the simpler approach- but rather wanted something with a better GUI and more user friendly. He decided to use the image component and have the user simply touch the image of the facial part he wants to change; that act would cycle through the available images. He further made use of the shaking of the phone which would provide a random selection for all facial parts. This showed how this student understood the importance of certain HCI concepts, which are often lost in introductory CS courses.

Service application was another theme required for a project. One student created an emergency application she named ICE. A user would click on an emergency button on the phone's screen which triggers an event sending a message with the current GPS coordinates to an emergency number. This project exemplifies how having a tool that offers a lot to the students and provides them with vast possibilities for application ideas. A project like this would have been complicated for a novice programmer to create in a textual programming language or in the amount of time it took using AIA.

Many students have created games utilizing the many features in the G1 phone; such games included duck hunt, snake, and hang man. One game that stood out was a QR code scavenger hunt game, which the student creating it called Marauder. The student wanted to create a game where G1 phone users would be able to compete in finding QR codes physically distributed throughout campus. Each QR code found would add to the score of the player depending on the difficulty of finding that code. This game required the utilization of the phone's camera for scanning and the existence of a database universally accessed. The TinyWebDB component in AIA provided such a purpose. That student even researched how to create her own online database to connect to when using the TinyWebDB component and was able to setup a general database for her application; with this she went beyond what was actually required. This project was the one that most exemplified p-creativity. That student used the features that were learned in class in a way that differed from their original purpose.

Overall the projects the students submitted showed the ease at with which the students worked with AIA. They furthermore show how the entire learning experience was one that fostered creativity. More importantly, they showcased in their projects most

of the basic programming concepts covered in class; showing an understanding of such concepts and an ability to apply them.

4.4.3 Reflection papers

After each programming assignment, the students were asked to write a reflection paper evaluating and reflecting on their experience in general and within the assignments in particular. Many common themes and categories have been found in analyzing the resulting reflection papers. The researcher's observations of the participants throughout the semester were added appropriately. Participants were given random labels in order to give the reader a chance to track each participant in each section or category. The themes that recurred in the different reflection papers were creativity, collaboration, AIA, and the overall course experience that different students had throughout the semester.

Creativity. The reflection papers the students wrote included some comments on what they perceived as the creativity aspects of the course. Their definition here does not relate to that mentioned in this study. Students were allowed to use the term in their own ways. Around the middle of the semester, *Student A* commented on how he found the assignments demanded an aspect of creativity, "our projects require us to play around with the software and be creative with our solutions." Most of the class was able to create creative apps for their assignments. Early in the semester, however, some students expressed discomfort with the fact that for assignments they were only given general themes and not specific instructions. *Student C* corroborated this as he explained his thought process early in the semester, "I have been trying to come up with a really cool and useful app to make but it seems like most of the good ideas have already been

created." But later on, in the final project, *Student C* had shown creativity when he created his own function when a block component in AIA did not work the way he expected it to, he stated,

> I worked with this block and I could never get it to work, so I designed a little method that allowed the ball to bounce off the objects. The method I used to get the ball to bounce required the use and understanding of some trigonometry. (*Student C*)

Student B was also struggling with the openness of the assignments early in the semester, "I am a creative person, however in this class I feel like I can never think of an application to build." Similar to *Student C*, as time passed *Student B* became more comfortable with coming up with ideas for applications and creating them, "when starting my final application I decided to try something new and step out of my comfort zone by creating a more complicated application." A lot of other students showed such comfort later in the course as well. They wrote that if given more time they would have done more with their projects. *Student A* commented, "I've come up with some creative ideas and can't wait to work on future projects to see if I can continue to perform well in this class." *Student B* showed the same sentiment of wanting to do more in her final project even though she was "happy with the finished application" stating, "if I had more time I would probably have" added more features. This, furthermore, showed that students felt motivated and interested in what they were learning. These two factors are often associated with creativity [73, 74, 48, 52].

Collaboration. A group page was created for this course to help student communicate outside of the classroom and encourage collaboration. Most students used

the group page from time to time to communicate and ask for advice on their projects. *Student D* commented on how the use of the group page had an important role, "being able to use Gmail to send class members info concerning problems we maybe[*sic*] experiencing with our projects is a godsend in project problem solving." This as well signified a willingness to utilize and an appreciation of communication and soliciting help from others. By mid-semester, students showed initiative in soliciting help from other students and learning from them. What was more important was that they exhibited a level of comfort in doing so. There were many comments showing evidence of that in the reflection papers,

> I did however notice that I was not the only one who could[n't] get a list to clear, . . . [*Student I*] also could not figure it out. When she told us she couldn't get it to work, . . . [*Student J*] had some ideas but they still wouldn't work. (*Student B*)
>
> I have helped with other groups apps and tried to throw in ideas when I could. (*Student C*)
>
> The group that I showed this app to gave me ideas of how to make it work. . . . The group gave me an idea that I tried. (*Student E*)
>
> What I have learned . . . is everybody can work together to make an application successful. . . . there were things that were challenging to pull off and the rest of the class had ideas to make something work. (*Student F*)

Many students commented on how collaborating and working in groups helped in problem solving and completing assignments. *Student H* commented on his collaboration

with another student on his final project and how useful it was, “the best part about working on this project is that . . . [*Student A*] and I helped each other out. One of us would not know how to do something and the other would.” Early in the semester, *Student A* commented on how he enjoyed his experience with collaboration, “I’ve really enjoyed working with other people and trying to finish things together.” He further commented on collaborating with other students and how it helped in problem solving, “while working . . . our group ran into obstacles that we had to overcome. And each time we were able to come up with some kind of solution, even if it wasn’t the best one.” Even though that student had a rocky start he finished the course with a straight A. Another student, *Student K*, also commented along the same line on collaboration, “we had a little trouble getting the blocks in the blocks editor correct but we figured it out as a team. . . . It was a lot of fun making this application.” *Student F* commented on how collaboration helped him in his learning experience, “I have learned a lot more working in a group than I would have if I was working by myself.” *Student M* commented on how the course provided a collaborative atmosphere for work to be done with little time spent outside of the classroom, “out of the last two groups, we have only had to meet outside of class once. Our application has worked both times and it was a great success with little time spent on it.” Another student showed regret in not making use of the collaborative atmosphere this course provided, “I could have used a group to work on it with[, it being the final project]. But it was my own fault for not getting one together to do so” (*Student L*). As seen here, collaboration was repeatedly mentioned by the students as an integral factor in learning and problem solving.

Some students commented on the difficulties they faced with collaboration. *Student A* commented on the difficulties of finding the group dynamics each time he started with a group, "each group I've gotten to work with works well in different ways, and it can be tough figuring out how to work well together in just a couple of days." *Student N* commented on how collaboration impeded his learning, "in the second group that I was in . . . [*Student J*] kind of took over and did most of the actual building of the application." However, *Student N* felt he was able to provide some contribution,

> For a while we couldn't get our project to work and eventually I provided the idea . . . that was responsible for making our application work. . . . while I didn't get to work on the actual building comments as much as I would have like to, I still had some valuable input. (*Student N*).

Student O was having a similar issue, "I feel like I need to strengthen my ability to work better in groups. It seems to me that in groups there is always one person that takes over the group." This exemplified an issue that often arises in group work. Having to mesh such different personalities together can be difficult, however, that's the way things are in the real world. Students here are taught early on how to deal with these different personalities in a way that gets the job done. This would be a great asset for them in their future careers.

AIA. In reflecting on their assignments, students have commented on their experience working with AIA. *Student P* commented on how comfortable he was creating application using AIA, "I feel that am able to use the Bock editor and to create interesting apps in the design view. I feel that I am comfortable with the android phone and the blocks editor." Early in the semester, *Student M* found AIA to require little time

to learn, “over the past couple of weeks I feel that I have greatly extended my knowledge in using the App [Inventor for] Android.” That student however found working with the blocks themselves a bit complex at that time but was optimistic about understanding visual blocks programming, “my weaknesses’[*sic*] are obviously still with the blocks editor. . . . my goals for the course are to better my understanding of the blocks editor and improve and expand my application’s abilities.” As updates are being made to AIA, *Student M* was still having some difficulties with the blocks editor but was able to get the job done, “I was having some technical difficulties with my pc and the blocks editor but overall it went alright.”

Near the beginning of the semester, one student commented on her experience in testing the limits of AIA and finding it capable of creating the app she wanted to produce, “when I do think about something I feel like the app builder cannot create what I want but in reality it can” (*Student B*). She commented later in the semester on how comfortable she was working with AIA and how easy she found using its IDE, “I found it fairly easy to place the blocks together to make it work.” Three students decided to collaborate on the final project and work on an application together. One group member, *Student E*, explained how AIA had met their expectations and created a delightful experience, “the overall application was a success and we enjoyed this project. It turned out pretty close to what we expected.” The other two members of his group concurred. *Student M*, another member of the group, showed motivation and a desire to showcase their understanding of key concepts taught in class in their project, “we wanted to do something that would not only challenge us, but be very interesting and bring about some key concepts in the development environment.” *Student M* and *Student K* - the third and

last member of the group - also commented on how they would like to learn more CS concepts, specifically *Sprites*. They were interested and motivated enough to learn the concept themselves, "we began knowing little about sprites feature and had to experiment with them to figure out exactly how they worked." Generally, all three members of the group showed a desire to learn more and a willingness to spend more time working on their project in order to add more features to the application they created. Conversely, and after six weeks from the start of the semester, *Student L* was struggling with understanding how AIA works. He himself attributed his struggles to the lack of interest and motivation,

> I still do not have a great concept of how everything works with App Inventor. . . . I would give myself a B for my basic knowledge and willingness to learn, but also for my lack of effort and enthuthasism [*sic*] towards the applications inventing itself. (*Student L*).

At the end of the semester *Student L* showed a bit of effort but not enough to show a difference in progress, "I found that being motivated was useful. I became unmotivated a while back, and it took me a while to get back into the mindset to get it to work." Overall, students found AIA easy to use in spite of the constant updates and changes being made. They were comfortable enough to experiment with the visual block and create many functioning applications. Some students even noted how interested they were in AIA and how motivated they were in using it.

Student H, furthermore, was able to lean certain CS concepts during his work on an application using AIA. He found it easy to experiment with changing his app using AIA to accommodate better Human-Computer Interaction (HCI),

> When I first built the program I used the *music player*[italic added] block and this did not work like I wanted it too[*sic*], because for some reason the music player did not respond quickly enough if you wanted to repeat the same sound rapidly. I then decided that it would not work with the music player. I then went back and found the block *sound player* [italic added]. The sound player worked perfectly for rapid pressing of the buttons. (*Student H*)

That same student created the game of hangman as his final project. Even though he was able to make it work, he faced some difficulties. He however attributed those difficulties to the lack of planning and time management,

> one thing that I believe would have made this project go a lot smoother and be less stressful is that we should have mapped out each part of the project because we would get done with one thing and not [k]now where to go after that. (*Student H*)

With that he showed an understanding of the importance of such concepts as HCI and time management in programming and in CS; concepts not often learned in introductory CS.

In contrast, some students reflected on some of the drawbacks they found during their work with AIA. *Student I* commented on the shortage of documentation on the use of AIA, "if the App Inventor had a solid amount of reference material and examples, I would be doing better." It might be worth mentioning here that *Student I* was one of the students that showed great progress throughout the semester and excelled near the end.

Student D agreed on the need for more documentations and manuals but found AIA simple enough to understand through trial and error,

> I personally feel that I could improve using block editor if there was some type of course book to refer to when I have questions I need answers to. . . . some type of course literature book would have been so helpful for a beginner in computer science like myself. I did find the block editor definitions a great tool to use in helping to figure out what blocks to use for my project. Also trial and error with the blocks helped me learn what worked and what doesn't work with the block editor. (*Student D*)

This was due to AIA's drag and drop nature where blocks are only allowed to drop in their correct places. This allowed students to experiment with AIA and create the diverse array of applications they had.

Furthermore, at the time of the study changes to a project in AIA can only be made on one computer station at a time. This caused a hurdle in the progress of a group project and, as *Student B* felt, rendered the group mute,

> One thing I don't like about the group projects is how we cannot all work on the one computer, . . . one person seems to take the lead over the other group members. By this happening in class I feel I am not learning as much as I could when it comes to understanding the app inventor for Android. (*Student B*)

Not a lot of students complained about that. In fact, students often rotated the use of the computer among group members or at times created roles for each member in the group.

In addition, *Student A* was not so happy with AIA and felt in had limited capabilities, "I've got a couple ideas for next projects that I don't think I want to attempt in App Inventor because I don't want to have to scale anything back." As *Student A* explained further about his experience working with AIA on a project he showed a concern for deadlines which may have hindered their process of experimenting and active learning, "we should have definitely used a couple of processes to make it simpler . . . we were also more worried about getting our application to be complete more than cleaning it up." Also, students that had some programming background found AIA boring and trivial,

> I haven't actually used the app.[*sic*] Inventor outside of class. Honestly, I haven't really had the need to. It is too easy. I used to program my own games onto my old commodore 64 when I was 12. . . . I have missed a few classes. I'm just bored with the app. [*sic*] Inventor it would be different if we could have multiple screen applications, etc. (*Student R*)

The entire experience was not lost on him. He did enjoy the idea of creating applications for the G1 phone,

> I have learned a few things here though. I have learned that I may like having a cell phone after all. . . . It's not the phone part of the phone that I like, it's the applications! They are awesome! (*Student R*)

In general, most of these mentioned drawbacks did not hindered the students' progress in the class and are easily overcome with better planning and exploration of AIA.

Course experience. Following several students' reflections individually throughout this learning process may shed light on their overall experience in this course. This section follows those students' experiences from beginning to end of semester. At

the end of this section, conclusions were drawn from their journeys. Of these students was *Student R*, one of the older students in this course. Early in the semester, he was having a mixed reaction to the overall experimental approach,

> I thought that this class was going to be a lot more technical. . . . Anyway, I have had fun and I like the entire class. We all seem to get along with one another pretty well. Hopefully things will continue along the same path, we're getting pretty used to it. (*Student R*)

Even though *Student R* was doing well in the class he ended up dropping the course later in the middle of the semester. The reasons behind his drop out were not clear. He may have lost interest in the course; however he did mention how busy he was with his job and family, "I don't skip class because I like to sleep in. I have two kids and they demands a lot of my time."

Another student, *Student P* had a turbulent experience throughout the course. In the beginning he showed interest in the course,

> The past few weeks have been in a manner of speaking, an adventure in CS 116. I've never really had access to something like the G1 phone or app inventor before. . . . I've had some really great experiences not only with the people in my group but also with the technology I'm working with. (*Student P*).

He further commented on the benefits of collaboration in terms of character building, "I also believe that working in groups not only brings everyone's ideas together to come up with some very interesting app work, but it also gives a chance for members to work on their strengths and weaknesses." *Student P* seemed motivated in his comment that he

"will make sure to put forth the effort to earn that A and gain the knowledge." Yet, by mid semester he was struggling and missing a few classes and was not showing the progress other students were showing at that time. Unlike *Student R*, he did not want to drop out of the course and showed motivation to learn,

> I do not want to drop this class, that has not even crossed my mind. I however do not just want to limp through the class and come out with no knowledge on the course that I had just taken. . . . I have finally gotten all my classes in order and set out in front of me with what needs to get done and when. (*Student P*)

After that statement, that student showed quite a bit of progress. He started to attend the class regularly and participate. By the end of the semester *Student P* demonstrated an understanding of the concepts taught in class and was able to create an application involving some of the more complex components in AIA for his final project,

> Something that I found rather useful with this project was the use of image sprites. By learning there[*sic*] uses and capabilities I found a way to figure them to do what I wanted within the game. It was also useful for me to look at the whack-a-mole tutorial and get a feel to how I was going to manipulate each of the sprites and their movements. Once I learned the basic behind the controlling of the sprites, I was able to set each one where I wanted it and gave it specific purpose. (*Student P*)

Student P found the *design crits* very useful and commented in writing saying that "these discussions bring together all the ideas of the class and allow for many problems to be

solved and many improvements to be made." He found the course overall to be "an awesome experience." In the end, that student passed the course with flying colors.

Many students expressed a sense of finding the course useful. "I really like how this class is going," *Student M* commented, "and I hope to keep learning and expanding my knowledge while building these different applications." That student showed in this comment a desire to learn more in the course. That desire did not fade away with passing time. A few weeks later *Student M* explained how given time he "would have experimented with ways to make the program more user friendly." *Student M* was doing well and maintained the same level of progress throughout the semester. At the end of the semester, he commented on how the course and the methodology of active learning administered "was a great tool to becoming more familiar with the more advanced functions of app inventor." In the end, that student had a transforming experience enough to make him actually apply for a change of minor to CS.

Student C expressed having a positive experience in the course early in the semester, "I feel like I have learned a lot in this class." Conversely, *Student C* expressed a disappointment in the material taught in the course and how he "hope[d] to learn more about the construct of programming in general." Later in the semester, *Student C* showed interest and motivation when he did more than what was required; he created an additional project to his submitted final project. He heard another student was struggling with the implementation of the Snake game thus decided to attempt it himself, "I did not come up with the idea but when other people were having problems I wanted to see if there really was any easy way to do it." Even though he struggled with the implementation himself, he stuck to it and did not give up,

> My original method that I was trying to use did not work at all. So I found out rather quickly that it was not going to be easy but I still love the idea of a snake game so I wanted to continue making it. (*Student C*)

He was able to actively find a solution that would make the application work. *Student C* decided to strip the implementation to the basic concepts of the game and "to make it on a small scale" and he "got it to work."

For *Student B* this course was her first experience in programming, "I have never done any programming what so ever until this class." Early in the semester, *Student B* had shown a good comprehension of CS concepts. She, however, was suffering from lack of confidence in her abilities, "I would like to have more confidence in myself, so I can tackle more in-depth applications." She commented on her learning experience in this course and how it extended beyond programming, "during the class I have not only learned how to program and make applications I have also learned some critical thinking skills." By end of the semester that student showed more confidence and more comfort in exploring her abilities, "I decided to try something new and step out of my comfort zone by creating a more complicated application." *Student B* showed good comprehension and learning in for a novice programmer,

> I don't think any other knowledge would have helped me overcome the problems I ran into. All I had to do was sit back look through the blocks I had put and run through the logic then I would figure out what I had placed in the wrong place. (*Student B*)

Another female student showed increasing progress throughout the semester. Early in the semester, *Student I* started a bit weak in her projects, as she herself admitted, "I cannot think of anything substantial because I feel like everything has been done that can be done." She further showed reactions of a novice programmer as she explained the difficulty she faced working with AIA, "[it] is also difficult within App Inventor because it doesn't do everything I would like it to do and it does not do things the way I expect it to. . . . it's foreign territory for most of us." She still however showed some motivation, "all I can hope to do is to continue figuring things out and come up with an app that in the end I will be very proud of." By mid semester, *Student I* progressed and showed better understanding of the material through her submitted work and seemed more interested in the course, "it will be interesting to see what people will come up with." She herself felt more confident in her abilities, "I am confident that through this project I have shown I can stand along side most of my peers in my knowledge of App Inventor." *Student I* also showed an increase in her motivation to learn, "I hope to continue thinking of more applications that will further push what I know." At the end of the semester she kept her word and did herself proud; not only by the final project she submitted but also by the extra effort she did in providing a piece of knowledge to the rest of the students in the class, "I completed this task easily and found myself done ahead of time, so I had to do something to push myself further," she stated. *Student I* created a tutorial for the rest of the class on how to setup a general database for an application and included the documentation necessary to follow through. Doing this extra work on her final project demonstrated the interest and motivation she was experiencing.

A male student, with no programming experience as well, wrote about his experience throughout the course. *Student D* commented on how he came into this course with no expectations or experience, "the first day of class I had no idea what to expect or even how to do anything on the computer for this class." He commented on collaborating in groups help him learn, "thanks to the brilliant idea of putting us all in small groups Mr. Gestwicki. I can't express enough how this has helped me greatly to learn from others how this all works." He goes further to explain how his experience changed from early in the semester,

> I didn't enjoy CS116 at the beginning, but now after working with others and learning how to build projects myself I can't[sic] hardly wait to see what I can build next. The main thing is I'm not afraid to ask someone else like . . . [*Student L*] n . . . [*Student N*] for help and it has made a world of difference for me. (*Student D*)

As he stated, the course increased his comfort level in seeking help and collaborating with other students. By the end of the semester, that student showed how he was till benefiting from the collaborative atmosphere SBL provided, "[*Student L* and *Student F*] were wonderful in helping me along the way also Mrs. Khuloud." His overall experience in the course helped change his views and be more open to computer science, "Over all[*sic*] I truly enjoyed the challenges of the class and all the unlimited fixable possibilities of computer science."

A different student, *Student J*, seemed to be very motivated throughout the course. At one point in the semester, even though he was "still a little new to some of the programming concepts," he was working on three projects simultaneously. What is worth

mentioning here is that he was able to make them all work. *Student J* aced the course with what seemed to be little effort on his part.

At the end of the semester, *Student N* explained how the atmosphere this experimental course provided created a knowledge seeking environment,

> When I was creating this [final project] application I used some of the other student's insights to help me create this application. I discussed several problems that I was having trouble with . . . [*Student B*] because she was creating a very similar application as me. I also found . . . [*Student J*'s project] very useful when I was creating my application. . . . This application was rather difficult to create . . . I contacted several of the students in the class that had the greatest knowledge about app inventor for suggestions on how to overcome this problem. (*Student N*)

As previously mentioned, this experimental approach relied on students seeking the knowledge. Lectures were very few yet interjections and interference from the instructors were almost nonexistent. This showed how students were actively learning and benefitting from the SBL model when used as the main teaching methodology.

In looking at those individual experiences one can see the motivation and interest students had throughout this course. They exhibited an increase in interest in CS. They benefitted from collaboration not only in finding solutions but in building their character as well. They were able practice problem solving through active learning. Their experience with SBL as the main teaching methodology was a positive one. Students were able to learn the material with few lectures given.

The students' reflections, moreover, coincided with some of the results from the other data collection tools. Similar to the results from the mind maps, students had experienced development in skills that went beyond programming. Concurring with the results from the MSLQ, some students had experienced an increase in their comfort levels in seeking help from their peers and collaborating.

4.5 Summary

In CS116, the experimental course, three modified MSLQs were administered, an entry-MSLQ, a mid-MSLQ, and an exit-MSLQ. With the exception of Peer Learning, all scales rated in the high range in all three MSLQs. The results did not indicate any statistically significant difference between the three modified MSLQ's. In CS110, the traditional course, only two modified MSLQs were administered, an entry-MSLQ and an exit-MSLQ. Similarly, an independent t-test was administered and no statistically significant differences were found. A comparison was implemented between the exit-MSLQs administered in both CS116 and CS110. The results of the independent t-test showed participants in CS116 rated significantly higher in the Control of Learning Beliefs, Help Seeking, and Intrinsic Motivation scales. In general, participants in CS116 were more motivated and interested in the course, they were more comfortable seeking help, and that they would get a positive outcome to their efforts more than participants in CS110.

Two interviews were implemented in CS116, one at the beginning of the course and one at the end. Participation in the interviews was voluntary and anonymous. Only one student participated in the entry-interview while seven students responded to the exit-

interview. The entry-interview showed the common attitudes and misconceptions exhibited in non-CS students. The exit-interview showed that the respondents had a positive reaction to the experimental course. Almost all respondents expressed that they would not change anything about the design of the course. Most respondents had positive responses about using AIA; even though most respondents had no programming experience before taking this course. For most respondents working on the projects and collaborating with others were enjoyable as well as beneficial. They had positive attitudes towards CS and found it relevant to their major and their lives. However, most respondents still showed some misconceptions regarding CS as a discipline.

Participants in CS116 completed two mind maps as well, one early in the semester and the other at the end. Both mind maps were created with CS at the root drawing any connections they found relative to it. The goal was to study their mental image of CS and any changes that occur from the start to the end of the course. By the end of the semester, participants were able to make more significant connections to CS. Thus, they demonstrated an improvement in their understanding of CS as a discipline.

Alongside the MSLQ, interviews, and mind maps results, CS116 students' submitted work was analyzed in this study. Participants in CS116 were able to complete all given assignments and showed creativity in their work. They were able to understand the basic programming concepts taught in the course and to utilize AIA and create applications for their G1 phones. The majority of the students scored in the higher range of grades. Not one student failed the course. In the reflection papers, most participants found collaboration to be a helpful part of the course. Overall, participants found the

experimental course to provide an interactive environment that cultivated collaboration and motivation to learn.

5. DISCUSSION

This chapter presents deeper interpretations of the results. It focuses on answering the research questions of the study and the possible factors that contributed to these results. The chapter is organized around the two main pedagogic factors of this study and the research questions.

5.1 Use of SBL

The SBL community has been able to show improved student retention and performance through their pedagogic methods [e.g. 17, 16, 90, 42, 38]. They have also used interviews and the MSLQ to measure student motivation in the classroom [e.g. 42, 38]. However, most research so far has retained lectures as the main teaching methodology and added an SBL element to it [e.g. 90, 41, 42, 38]; such as a SBL lab [e.g. 25]. In this study, the SBL approach was used as the main teaching method to teach an introductory CS course; an approach rarely seen in the literature. Lectures were rarely used and when they were their topics were directed by the issues the students brought up in the *design crits*.

In this study, parallel to previous studies [17, 23, 25, 42, 41, 38], participants have responded favorably to SBL activities. Participants have enjoyed the *design crits,* a key

activity of SBL, and found them to be an integral part of their learning process. A participant's statement culminated the anticipated benefits of them when he stated that "these discussions bring together all the ideas of the class and allow for many problems to be solved and many improvements to be made." These results are in agreement with the findings of Carbone and Sheard [16] and that of Gottel and Schild [30] where students saw the usefulness of the *design crits* to their learning process. Outside of the *design crits*, students must depend on themselves to find solutions to given problems. The SBL model allows them to seek help, have discussions, and even collaborate with others in order to do so. Through their reflections and interviews, the participants revealed a belief that these collaborative activities helped them succeed and complete their projects. This was evident in the MSLQ's Control of Learning Beliefs scale as well, which indicated that participants attending the experimental course believed that their efforts to learn will result in positive outcomes significantly more than participants attending the traditional course. These results suggest that SBL, with its collaborative atmosphere, provided a space for students to actively learn. This was evident in almost all the data collection tools utilized in this study. Through the reflection papers and interview responses, participants attending the experimental course showed an appreciation and an ease in collaborating and learning from others and how such collaboration helped in problem solving and completing assignments. In the MSLQ, participants scored significantly higher in the Help Seeking scale in the experimental course; indicating they interacted more with others when needed than participants attending the traditional course. In fact, participants attending the traditional course scored very low rates when it came to peer learning and help seeking. It was evident that collaboration was nonexistent

in the traditional course, as far as the participants were concerned. Actually, collaboration and peer learning were found to be an integral part of the experimental course. This is similar to the findings of a previous study by Forte and Guzdial [27]. Some participants went as far as making them parts of CS as a discipline, according to the mind maps results. Such results confirmed the effect of SBL and its suitability for introductory CS courses as a teaching methodology.

Collaboration, though, is not without its faults. One must be aware of personality problems, where dominant personalities tend to take over and do all the work while the rest of the group watch, as some participants had experienced in this course and commented on in their reflection papers. Participants, furthermore, have to learn to deal with the clash of personalities that often arises in collaborative work; which can be character building and a great skill for their future careers. This is parallel to the findings of Hundhausen et al. [40] that SBL helps students acquire valuable skills for their future profession.

In general, the results are indicative of the atmosphere provided by SBL, as a factor in this experimental approach, being more supportive of the learning process. This supports Estey et al. [25] conclusion that students benefited from the different perspectives and sense of community SBL provides. The results also imply that SBL can effectively be used as the main teaching methodology in an introductory CS Course.

5.2 Use of AIA

This study, also, evaluated AIA as a teaching tool. Ball State University was one of thirteen institutions invited to participate in this closed beta evaluation of AIA,

specifically of its applicability in introductory computer science courses for non-CS majors. The software has the potential for significant impact and widespread adoption, partially due to its backing by both Google and renowned computer science professor Hal Abelson; Also due to it providing an easy way of creating mobile application for novice programmers in a simple drag and drop tool.

At the time of the study AIA was still under development. In the beginning, working with AIA was found to be a bit challenging for both the students and the instructors. Even though AIA started with limited functionalities and blocks library, these functionalities made use of the complex features that the G1 phone provides, such as GPS and G-sensor. Not surprisingly, perhaps, other studies have found it challenging to work with tools that have limited functionalities and libraries [55, 50, 33]; yet those challenges were overcome and the results were favorable. Throughout the semester, and based on recommendations and requests from participating universities evaluating AIA, continuous changes and additions were made by the AIA development team at Google to incorporate more new features and functionalities. While this had provided students with new features and expanded capabilities, it created some challenges for the students that at times resulted in frustration, as reflected in some of the participants' comments and observed by the researcher. By the time students were working on their final project, and thanks to the diligent work of the AIA development team, AIA had a diverse library of features and functionalities for students to use. Overall, participants still expressed positive comments about the use of AIA as the programming language for this introductory course. The challenges of it being under development were overcome and the results were favorable here as well.

Students in introductory CS courses are often confused and don't know what to create in terms of applications. They are often, as well, distracted with their struggles with learning the textual syntax of the programming language used [40]. The interviews as well as participants' reflections indicated that what they liked most about AIA is its ease of use. It allowed them to create mobile applications in a drag and drop environment without the struggles that occur in using textual syntax; "It did make the thought process way easier," as one participant commented. It, furthermore, allowed the participants to be creative and create all sorts of applications that make use of the different functions of the phone while at the same time keeping them focused on the fundamental concepts in CS. For example, within a few weeks from the beginning of the course participants were able to created applications that make use of the shake event of the phone without having to go into the details behind it, details that are usually taught in a more advanced CS course not introductory CS. This gave the participants the ability to create complicated applications using simple drag and drop operations while only needing to understanding fundamental concepts in CS such as loops and conditionals. This was most evident in the applications participants created throughout the semester. Using AIA eliminated the need to spend time at the beginning of a course on learning tedious, but necessary, details before actually being able to create a substantial application. This addressed the concerns Forte and Guzdial [27] had about creating an educational environment that allowed students to learn through taking advantage of the features an unfamiliar media provides "without first investing immoderate amounts of time learning to program" (p. 2). In this course, participants have created all types of apps while showing creativity with their choice of applications and the media they used in terms of sounds and images. What

made this even more remarkable is the fact that they had no prior programming experience.

One concern of this implementation might be the cost of the G1 phones. Use of the phones played a major role in motivating students by providing the immediate reward of seeing their work first hand and in context. Hopefully, with the cost of such advanced mobile devices going down as the technology advances, this will be less of an issue in the future.

5.3 The Effect of This Experimental Approach

On motivation. The experimental course, with its two pedagogic factors (AIA and SBL), have shown an effect on motivation. According to MSLQ results, participants attending that course exhibited significantly high interest and motivation levels, even significantly higher than participants attending the traditional course. This may indicate the positive effect this approach has on motivation to learn in general. In terms of reasons behind the motivation, participants in the experimental course showed higher motivation due to intrinsic aspects than participants in the traditional course. Such results are indicative of a significantly higher interest in the experimental course itself and the CS concepts taught in it, as well as a significantly higher engagement in the learning process. Thus, the experimental approach sparked more interest in those attending it, which may be attributed to the use of a programming language that produced mobile applications. Participants found working with AIA and creating mobile applications interesting. This was most evident in their desire to create more applications and even add more features to finished projects even after submission, when they were no longer required to; which

was emphasized often in their reflection papers and interview responses. Most prominent in the participants' statements, moreover, was the instant gratification they got from creating the mobile applications and using them on their phones, thus confirming Kurkovsky's [50] claim. Similar to previous studies [54, 55, 50, 72], these results confirm the effect of using mobile application development to inspire, interest, and motivate students to learn.

On creativity. In the experimental course, participants have expressed that the course gave them an outlet for creativity. Participants have produced applications for their mobile phones that showed a sense of P-creativity (e.g. Pumpkin Head App, Marauder). The fact that non-CS students, with no prior programming experience, attending an introductory CS course were able to create functioning applications is in itself remarkable. Similar to Guzdial's media comp approach [33], participants in this experimental approach were "taking advantage of the creative aspects of the course" (p. 6) [27] and were coming up with interesting ideas and producing creative applications.

It was noted, however, that at the beginning of the course participants were uncomfortable with the openness of the assignments and having to realize their own ideas; participants have reflected on this issue in their early reflection papers. This supports Gottel's and Schild's [30] findings that students are "not well-trained in creative thinking" (p. 98). As the course progressed, however, participants became more comfortable with the openness of the assignments and were able to come up with many creative applications. This may suggest the usefulness of SBL as a training environment

for creative thinking and the importance of a user friendly programming language such as AIA in facilitating it.

As Romeike [73] established, creative activities increase students' motivation. MSLQ results have shown significantly higher levels of intrinsic motivation in the experimental course than the traditional, thus the participants in that course must have experienced higher levels of creative activities. Romeike [74], furthermore, defined three dimensions of creativity that have a significant impact on the creativity process in CS education. This experimental course has covered all three dimensions of creativity that Romeike [74] defined. The PERSON-dimension was covered in the cultivation of intrinsic motivation as shown in the MSLQ results. The SUBJECT-dimension was covered in the division of the *design crits* into *Studio, Pitch, and Presentation* as the creative process of application development throughout the course and the participants' created applications. Finally, the ENVIRONMENT-dimension was the use of AIA and its support of creativity in its ease of use along with facilitating complicated features of the G1 phone for implementation in an introductory CS course.

On comprehension. In the experimental course, participants' comments and reflections indicated that they did not find difficulty in understanding the material taught and in implementing it. They indicated an increase in their ability and willingness to understand the material as the course progressed. In fact, the MSLQ results implied that, as the course progressed, these participants found it was becoming significantly easier. In this course, most of the problems participants faced in creating their applications related to errors in logic. The use of AIA eliminated the distractions that often occur in the use of textual programming languages and helped students focus on algorithmic comprehension.

When a participant had a problem with his application it did not involve a missing semicolon or comma. It was usually him misunderstanding a concept thus giving the instructors the chance to correct that misconception leading to a better understanding of the logic that the participant was trying to achieve in his program. In the first half the semester, participants struggled with the logic behind some of the simpler basics of programming, such as string concatenations and conditionals. As the course progressed, however, they were able to understand such concepts and the process became easier to them. Their submitted programming assignments and assessment processes corroborated this as well, where participants have shown an understanding of CS concepts and the basic principles presented in class. Debugging overall took less time in using AIA and participants seemed to focus more on design issues as the course progressed. These results are in agreement with the findings of Hundhausen et al. [41] and Wilcox et al. [87].

In the experimental course, participants showed an understanding of specific concepts related to CS that are usually either difficult to understand at that stage or even not at all introduced in the material taught in such an introductory CS course. One concept that is often not usually introduced in introductory CS courses, but was naturally introduced here, was planning and managing projects. Due to design of the course, with the existence of *design crits*, students must do product presentations and abide by deadlines given otherwise they miss the opportunity to display their apps to the rest of the class and get feedback; as one participant commented: “I made sure that I picked a project that was doable in that amount of time and also was useful.” Marketability was another concept that showed in the participants’ work and comments; as one participant

stated when reflecting on his choice of project that he wanted to create "something useful that a person might actually use more than once." Yet another concept was usage design; specifically the difference between the two concepts of What You See Is What You Get verses What You See Is What You Need (WYSIWYG vs. WYSIWYN). WYSIWYN revolves around eliminating distractions for the user of the application and only providing the visible elements needed to accomplish the task at hand. Participants' projects and their submitted work showed their understanding of this concept and which GUI components to use; which components would provide better usage and simplify the understanding of how the app works, making the process more intuitive to the user. In their reflection papers, the participants have shown their understanding of the importance of such design concepts.

On achievement. Unlike other courses, students only received two grades throughout the semester: the midterm and the final grades. This experimental course depended mostly on formative assessment, both formal and informal, where students received constant feedback. The midterm grade was more of an indicator of the students' progress and achievement levels at that time. The aim was to inform them of their progress so far and whether they needed to work harder and put more effort into the course, many of them did. At the end of the semester, the MSLQ results indicated that participants attending the experimental course expressed significantly more faith in getting a positive outcome than those attending the traditional course. Participants attending the experimental course were able to complete all given assignment. Almost half of the participants got an 'A' and not one participant failed the course. For each participant, the final grade represented the cumulative work and the progress throughout

the semester. This success may be contributed to AIA's drag and drop environment that allowed students to experiment. It may be attributed to SBL's supportive atmosphere that accommodated discussions and collaboration. There was, however, no clear indication to which of these factors contributed more to the participants' achievement.

On comfort level. Participants in the experimental course found it to provide a comfortable learning environment. They often commented on being comfortable in collaborating with other students and asking for help. In fact, the results of the MSLQ showed that those participants were significantly more comfortable collaborating with and seeking help from other students than participant attending the traditional course. This is similar to the finding of a previous study by Hundhausen et al. [42]: an increase was found in the comfort level in collaboration for students attending the SBL course. In the exit-interview, as well, participants commented on the benefits of the social aspect of the model; "I was more willing to reach out to people for help," one participant commented. These results call attention to the social environment provided by SBL, with its *design crits* and collaboration, where participants were more comfortable interacting with and learning from each other. As concluded by Burgin and Reilly [14] and by Wilson and Shrock [88], this comfort level translated into their performance where the majority of the participants in the experimental course did very well with almost half of them acing the course; moreover, they were able to complete all given programming assignments.

On attitude. With regards to attitudes towards the experimental course itself, the MSLQ results indicated that participants in the experimental group showed significantly more interest, felt significantly more comfortable, and were significantly more likely to

recommend this experimental course to other students; more than participants in the traditional group recommending the traditional course. Participants attending the experimental course found it to meet their expectations, as a pedagogy and environment, and found its approach to be satisfactory. One participant went as far as describing his experience as an "immersive learning experience." Even though this course does not follow the BSU's definition of immersive learning, it is still significant that that participant felt that way. Moreover, participant often commented in their reflection papers and essays on how they enjoyed both the use of AIA as the learning tool and the use of SBL as the learning model. Such results suggest a positive attitude towards this course and its pedagogic factors.

With regards to attitudes towards CS, participants attending the experimental course have indicated that they would take yet another CS course, as the MSLQ results showed. Moreover, the MSLQ showed that they were significantly more likely to take another CS course than participants in the traditional group. This may imply that they enjoyed the course enough to be encouraged to take another one. The results of the mind maps analysis suggested that participants attending the experimental course exhibited a statistically significant change in their understanding of CS. It no longer evolved only around programming. This was also evident in their reflection papers, where participants have shown an understanding as well as an appreciation of such non-programming aspects. This change of view may be attributed to the introduction of advanced topics that allowed the student to be exposed to related CS concepts that went beyond programming. This was made possible by AIA which allowed the time to cover such advanced concepts by eliminating distractions of textual syntax and requiring little time to learn. Challenging

this result was one found in the exit-interview, where five respondents to that interview showed some misunderstanding of CS. However, other than these five responses by these five respondents in this one interview, all other data collected indicated an understanding of CS. Another change of view was exhibited in this study; the image of those majoring in CS seemed less nerdy, as one participant commented: "I looked at computer science as something only nerds could do but it is actually rather simple." This change of view may be attributed to the social environment provided by SBL. The most significant result in this study was the desire of two participants from the experimental course to change their minors to CS all together. Collectively, these results imply that this experimental course had a positive effect on attitudes towards CS.

5.4 Further Research

A follow up study with the students that participated in this study should be conducted to investigate whether this class has helped them in other CS related courses. Malan and Leitner [56] claimed that students would transition successfully to a textual syntax language like Java. Therefore, a study is needed to investigate the participants' transition to a textual syntax language and their ability to assimilate what they have learned to new situations. Also investigate whether this experimental approach has helped them learn other programming languages they have come across since they took this experimental course.

A repeat of this study with larger number of participants is recommended in order to confirm the results of this study and allow for increased generalizability. One particular aspect of the study may raise an important question: would the same results

have merged with different instructors? The possibility exists that teacher's personality may have influenced participants and the results of this study. However, the fact that participants' comments throughout the study lacked any mention or indication of teachers influence, combined with the use of many data tools that focused on students and their interaction with the factors of this experimental approach, would seem to mitigate this concern. Nonetheless, it would be interesting to include this variable in future research. Another aspect that may raise another important question: would implementing this experimental approach in an introductory CS course with CS-majors increase retention? The results indicate that CS-majors might find the course interesting and motivating but is it enough to increase retention in CS as a major. This study did show that students in the experimental course would take more CS courses; which may be an indication of possible retention. However, further study in needed to confirm the effect this approach has on retention of CS-majors. Additionally, testing the difference in response of the two gender to this approach would also be interesting; specifically female students. CS as a discipline is less appealing to females than males [1,57,27,81]. Studying the effect of this approach on female students, might help shed light into ways to attract female students to the CS major.

A more rigorous comparison is recommended between this experimental approach and a traditional introductory CS course that relies on lectures and a text-based instructional language. This study would also include implementing all the data tools used, not just the MSLQ, on students attending the traditional course. Using the mind maps would allow a comparison in the change of view of CS as well as attitude toward CS between the two teaching methodologies. Studying the difference in achievement

would shed light on the effect this approach has as well. Such a study would also investigate whether students attending the traditional course would have had more or less trouble in the text-based environment.

5.5 Conclusions

Recently the CS major has been experiencing a slight but steady increase in retention. This increase in retention may be due to all the resent research on reinventing introductory CS courses. Many studies have implemented many different methodologies of teaching in introductory CS for both majors and non-majors with favorable results. Indicating that such trends of change in the CS curriculum might be just what the major needs to rejuvenate it and bring it back to its glory days.

As mentioned previously, introductory CS courses have notoriously low success rates despite consistent demand from the industry for CS graduates. This study implemented an experimental introductory CS course for non-CS majors focusing on two pedagogic factors, SBL and visual blocks programming for mobile applications– specifically AIA. It was conducted with the hopes of improving our understanding as educators of non-CS students' motivation in and perception of introductory CS courses.

The use of SBL as the main teaching methodology is rarely found in the literature. In this implementation, only five mini lectures were given throughout the semester and were guided by the *design crits*. The instructors saw little need to interject or interfere in the learning process. Students were able to learn on their own and, at the same time, have a positive attitude about the course. In this experimental course, students scored in the high range on all the scales of the MSLQ. This indicated that they were doing well in

terms of motivation and learning skills. The results of this study, furthermore, showed that students attending this experimental course had higher motivation, collaboration, and comfort levels. Compared to the traditional course, students attending the experimental course were more optimistic about their learning process resulting in positive outcomes, and had a higher desire to collaborate with their peers and seek assistance when needed.

The experimental course as a whole was found to be a positive experience. Creating mobile applications as part of the course was found to be one of the factors that contributed to increasing motivation. Basing the pedagogy on SBL was a factor in increasing collaboration and comfort levels. Both factors resulted in positive outcomes in terms of students' motivation, performance, and attitude. Students were able to pickup basic concepts and actively learn through this course. Attitudes toward CS were affected as well. This study indicated that this experimental approach resulted in positive attitudes towards the discipline. The students' image of CS as a discipline was effected by this experimental approach as well. Students, at the end of the semester, no longer acquainted it with programming alone. Students were able to appreciate other factors related to the discipline, such as design and HCI issues. The anti-social image of the discipline was also affected. They found CS inviting, challenging, and interesting. In both mind maps and interview results, students associated collaboration with CS. This result appears to confirm previous research by Hundhausen et al. [40]: the anti-social image the discipline has is due to associating computing with programming and tending to focus mainly on learning syntax of a language in early CS course. In Fact, students in the experimental course were found more likely to take more CS courses. Consequently, this approach had a deeper effect on a couple of students which resulted in their desire to change their

minor to CS. This conclusion is in agreement with a previous study by Lewis et al. [53] in which the researchers found that there are five factors that influence students to major in CS: ability, enjoyment, fit, utility, and opportunity cost. In this situation, the opportunity cost factor caused the students to only change their minor, since they already have declared majors and were well into their study plans where changes at this point would not be practical.

In conclusion, this study suggests that a course merging a visual blocks language, with the use of a medium that is of interest to students at the time of study–in this case mobile technology, along with the use of an active learning pedagogy, creates a learning environment that cultures creativity and trains student in creative thinking. It, furthermore, shows that SBL can stand on its own as the main teaching methodology used in an introductory CS course. Even though the sample size is relatively small, this study provides validation and new perspectives on the literature on the use of SBL, as the main teaching methodology, and visual blocks programming languages in Computer Science. Furthermore, the analysis of data collected in this study may help to improve recruitment and retention of CS majors in addition to improving the understanding of how to make CS accessible and interesting to non-majors.

REFERENCES

[1] AAUW. *Tech-Savvy: Educating Girls in the New Computer Age.* American Association of University Women Education Foundation, New York, 2000.

[2] ACM/IEEE-CS Joint Task Force. Computing Curricula 2001: Computer Science. *Journal on Educational Resources in Computing (JERIC)*, *1*(3), 2001.

[3] ACM/IEEE-CS Joint Task Force. *Computer Science Curriculum 2008: An Interim Revision of CS 2001*. 2008, Retrieved from http://www.acm.org/education/curricula/ComputerScience2008.pdf

[4] Guy-Alain Amoussou, Eileen Cashman, and Steve Stienberg. Ways to Learn and Teach Creativity and Design in Computing Science. *Proceedings of the 2007 Symposium on Science of Design,* 12-13, 2007.

[5] App Inventor Fall 09 Pilot. https://sites.google.com/site/androidblocks/

[6] Alice. http://www.alice.org/

[7] Mordechai Ben-Ari. Constructivism in Computer Science Education. *Journal of Computers in Mathematics and Science Teaching, 20*(1): 45-73, 2001.

[8] Margaret A. Boden. *The Creative Mind: Myths and Mechanisms,* 2nd ed. Routledge, New York,2004.

[9] John Seely Brown. New Learning Environments for the 21st Century: Exploring the Edge. *Change, 38*(5): 18-24, 2006.

[10] Jerome S. Bruner. *The Process of Education.* Harvard University Press, Cambridge, MA, 1960.

[11] Bureau of Labor Statistics. Computer Programmers. *Occupational Outlook Handbook, 2008-2009 Edition,* 2009.

[12] Bureau of Labor Statistics. Computer Programmers. *Occupational Outlook Handbook, 2012-2013 Edition,* 2012, retrieved on 5/23/2012 form http://www.bls.gov/ooh/computer-and-information-technology/computer-programmers.htm

[13] Bureau of Labor Statistics. Tomorrow's Jobs. *Occupational Outlook Handbook, 2008-2009 Edition,* 2009.

[14] Susan Burgin and Ronan Reilly. Programming: Factors that Influence Success. *ACM SIGCSE Bulletin*, *37*(1): 411–415, 2005.

[15] Tony Buzan and Barry Buzan. T*he Mind Map Book: How to Use Radiant Thinking to Maximize Your Brain's Untapped Potential*. Penguin Books, NY, 1994.

[16] Angela Carbone and Judy Sheard. A studio-based teaching and learning model in it: what do first year students think?. *Proceedings of the 7th annual conference on Innovation and technology in computer science education*, 213–217, 2002.

[17] Adam S. Carter and Christopher D. Hundhausen. A Review of Studio-Based Learning in Computer Science. *Journal of Computing Sciences in Colleges, 27*(1), 2011.

[18] J. McGrath Cohoon and Lih-Yuan Chen. Migrating Out of Computer Science. *Computing Research News, 15*(2): 2-3, 2003.

[19] Steve Cooper and Steve Cunningham. Teaching Computer Science in Context. *ACM Inroads*, *1*(1): 5-8, 2010.

[20] Tim DeClue. A Theory of Attrition in Computer Science Education which explores the Effect of Learning Theory, Gender, and Context. *CCSC Central Plains*, ,2009.

[21] Peter J. Denning. Great Principles of Computing. *Communications of the ACM, 46*(11): 15-20, 2003.

[22] John Dewey. *Democracy and Education.* Macmillan, New York, 1958.

[23] Michael Docherty, Peter Sutton, Margot Brereton, and Simon Kaplan. An Innovative Design and Studio-based CS Degree. *ACM SIGCSE Bulletin,33*(1): 233-237, 2001.

[24] Mark Dunlop and Stephen Brewster. The Challenge of Mobile Devices for Human Computer Interaction. *Personal and Ubiquitous Computing, 6*, 253-256, 2002.

[25] Anthony Estey, Jeremy Long, Bruce Gooch, and Amy A. Gooch. Investigating Studio-based Learning in a Course on Game Design. *Proceedings of the Fifth International Conference on the Foundations of Digital Games,* 64-71, 2010.

[26] L. Dee Fink. *Creating Significant Learning Experiences: An Integral Approach to Designing College Courses*. Joosey-Bass, San Francisco, 2003.

[27] Andrea Forte, and Mark Guzdial. Computers for Communication, Not Calculation: Media as a Motivation and Context for Learning. *Proceedings of the 37th Annual Hawaii International Conference on System Sciences*, 1- 10, 2004.

[28] Eric Freudenthal, Mary Roy, Alexandria Ogrey, Tanja Magoc, and Alan Siegel. MPCT - Media Propelled Computational Thinking. *Proceedings of the 41st ACM technical symposium on Computer science education*, 37-41, 2010.

[29] Timo Göttel. Virtual Sandbox – Adding Groupware Abilities to *Scratch. Proceedings of the 8th International Conference on Interaction Design and Children,* 158-161, 2009.

[30] Timo Gottel and Jonas Schild. Creativity Room 5555: Evoking Creativity in Game Design Amongst CS Students. *Proceedings of the 16th annual joint conference on Innovation and technology in computer science education*, 98-102, 2011.

[31] Paul Gross and Kris Powers. Evaluating Assessments of Novice Programming Environments. *Proceedings of the first international workshop on Computing education research,* 99–110, 2005.

[32] Mark Guzdial. Use of Collaborative Multimedia in Computer Science Classes. *Proceedings of the 6th annual conference on Innovation and technology in computer science education*, 17-20, 2001.

[33] Mark Guzdial. A Media Computation Course for Non-Majors. *Proceedings of the 8th annual conference on Innovation and technology in computer science education*, *35*(3): 104-108, 2003.

[34] Mark Guzdial. *Introduction to Media Computation: A Multimedia Cookbook in Python*. Pearson, Boston, MA, 2004

[35] Mark Guzdial and Barbara Ericson. *Introduction to Computing & Programming in Java: A Multimedia Approach*. Pearson Prentice Hall, 2007.

[36] William Hansen. A Report on the Use of Multimedia Courses in Computer Science Education. *Papers of the SIGCSE/CSA Technical Symposium on Computer Science Education,* 35, 1978.

[37] Richard Henderson. Industry employment and output projections to 2020. *Monthly Labor Review, 135*(1): 65-83, 2012.

[38] Dean Hendrix, Lakshman Myneni, Hari Narayanan, and Margaret Ross. Implementing Studio-Based Learning in CS2. Proceedings of the 41st ACM technical symposium on Computer science education, 505-509, 2010.

[39] Leslie S. Hiraoka. *Underwriting the Internet: How Technical Advances, Financial Engineering, and Entrepreneurial Genius are Building the Information Highway*. M.E. Sharpe Inc., Armonk, New York, 2005.

[40] Christopher D. Hundhausen, N. Hari Narayanan, and Martha E. Crosby. Exploring studio-based instructional models for computing education. *Proceedings of the 39th SIGCSE technical symposium on Computer science education*, 392–396, 2008.

[41] Christopher D. Hundhausen, Anukrati Agrawal, Dana Fairbrother, and Michael Trevisan. Integrating Pedagogical Code Reviews into a CS 1 Course: An Empirical Study. Proceedings of the 40th ACM technical symposium on Computer science education, 291-295, 2009.

[42] Christopher D. Hundhausen, Anukrati Agrawal, Dana Fairbrother, and Michael Trevisan. Does Studio-Based Instruction Work in CS 1?: An Empirical Comparison with a Traditional Approach. Proceedings of the 41st ACM technical symposium on Computer science education, 500-504, 2010.

[43] Andy Hunt. *Pragmatic Thinking and Learning: Refactor Your "Wetware"*. The Pragmatic Bookshelf, Dallas, TX, 2008.

[44] Jython Project. http://www.jython.org/Project/

[45] Walter Kintsch. The Role of Knowledge in Discourse Processing: A Construction-Integration Model. *Psychological Review, 95*(2): 163-182, 1988.

[46]Walter Kintsch. *Comprehension: A paradigm for cognition*. Cambridge University Press, Cambridge, UK, 1998.

[47] Walter Kintsch and Teun A. van Dijk. Toward a Model of Text Comprehension and Production. *Psychological Review, 85* (5): 363-394, 1978.

[48] Maria Knobelsdorf and Ralf Romeike. Creativity as a Pathway to Computer Science. *Proceedings of the 13th annual conference on Innovation and technology in computer science education,* 286-290, 2008.

[49] Robert B. Kozma. Will Media Influence Learning? Reframing the Debate. *Educational Technology, Research and Department, EARLI Conference,* 2:1-31, 1993.

[50] Stan Kurkovsky. Engaging students through mobile game development. *Proceedings of the 40th ACM technical symposium on Computer science education*, 44–48, 2009.

[51] Paul D. Leedy and Jeanne E. Ormrod. *Practical Research: Planning and Design,*8th ed. Pearson, Columbus, OH, 2005.

[52] Gary Lewandowski, Elizabeth Johnson, and Michael Goldweber. Fostering a Creative Interest in Computer Science. *Proceedings of the 36th SIGCSE technical symposium on Computer science education*, 535–539, 2005.

[53] Colleen M. Lewis, Ken Yasuhara, and Ruth E. Anderson. Deciding to Major in Computer Science: A Grounded Theory of Students' Self-Assessment of Ability. *Proceedings of the Seventh International Workshop on Computing Education Research*, 3-10, 2011.

[54] Quasy H. Mahmoud and Allan Dyer. Integrating BlackBerry Wireless Devices into Computer Programming and Literacy Courses. *Proceedings of the 45th annual southeast regional conference*, 495-500, 2007.

[55] Quasy H. Mahmoud and Allan Dyer. Mobile Devices in an Introductory Programming Course. *Computer, 41*(6): 108-107, 2008.

[56] David J. Malan and Henry H. Leitner. Scratch for Budding Computer Scientists. *Proceedings of the 38th SIGCSE technical symposium on Computer science education*, 223-227, 2007.

[57] Jargolis Margolis and Allan Fisher. *Unlocking the Clubhouse: Women in Computing.* The MIT Press, Cambrige, MA, 2002.

[58] John R. McClure and P.E. Bell. *Effects of an Environmental Education Related STS Approach Instruction on Cognitive Structures of Pre-Service Science Teachers*. Pennsylvania State University, University Park, PA, 1990.

[59] John R. McClure, Brian Sonak, and Hoi K. Suen. Concept Map Assessment of Classroom Learning: Reliability, Validity, and Logistical Practicality. *Journal of Research in Science Teaching, 36*(4): 475–492, 1999.

[60] Wilbert J. McKeachie and Marilla Svinicki. *McKeachie's Teaching Tips: Strategies, Research, and Theory for College Students and University Teachers.* Houghton Mifflin Company, New York, 2006.

[61] Sharan B. Merriam. *Qualitative Research: A Guide to Design and Implementation*. Jossy-Bass, San Francisco, 2009

[62]R. Mark Meyer and Tim Masterson. Towards a better visual programming language: critiquing Prograph's control structures. *Proceedings of the fifth annual CCSC northeastern conference on The journal of computing in small colleges*, 181–193, 2000.

[63] Paul Mullins, Deborah Whitfield, and Michael Conlon. Using Alice 2.0 as a First Language. *Journal of Computing Sciences in Colleges, 24*(3): 136-143, 2009.

[64] Gayle Nicoll. A three-tier system for assessing concept map links: a methodological study. *International Journal of Science Education, 23*(8): 863- 875, 2001.

[65] Joseph D. Novak and D. Bob Gowin. *Learning how to learn*. Cambridge Press, New York, 1984.

[66] OpenBlocks. http://dspace.mit.edu/handle/1721.1/41550

[67] N. Renee Pearsall, Jo El J. Skipper, and Joel J. Mintzes. Knowledge restructuring in the life sciences: a longitudinal study of conceptual change in biology. *Science Education, 81*(2): 193-215, 1997.

[68] Paul R. Pintrich, David A. F. Smith, Teresa Garcia, and Wilbert J. McKeachie. *A Manual for the Use of the Motivated Strategies for Learning Questionnaire (MSLQ).*The University of Michigan, 1991.

[69] Atanas Radenski. "Python First": A Lab-Based Digital Introduction to Computer Science. *Proceedings of the 11th annual SIGCSE conference on Innovation and technology in computer science education,*197-201, 2006.

[70] Cecil R. Reynolds, Ronald B. Livingston, and Victor Willson. *Measurement and Assessment in Education,* 2nd ed. Pearson, New Jersey, 2009.

[71] Lauren Rich, Heather Perry, and Mark Guzdial. A CS1 Course Designed to Address Interests of Women. *Proceedings of the 35th SIGCSE technical symposium on Computer science education*, 190–194, 2004.

[72] Christian Robenson. Aliening Generations to Improve Retention in Introductory Computing Courses. *Journal of Computing Sciences in Colleges, 26*(6): 30-36, 2011.

[73] Ralf Romeike. Creative Student – What can we Learn from Them for Teaching Computer Science?. *Proceedings of the 6th Baltic Sea conference on Computing education research*, 149-150, 2006.

[74] Ralf Romeike. Three Drivers for Creativity in Computer Science Education. In *Proc. of the IFIP-Conference on "Informatics, Mathematics and ICT: a golden triangle"*. Boston, USA, 2007.

[75] Robert Rotenberg. *The Art & Craft of College Teaching: A Guide for New Professors & Graduate Students.* Left Coast Press inc, Walnut Creek, CA, 2005.

[76] Nathan Rountree, Janet Rountree, Anthony Rountree, and Robert Hannah. Interacting Factors that Predict Success and Failure in a CS1 Course. *ACM SIGCSE Bulletin*, *36*(4): 101–104, 2004.

[77] Maria A. Ruiz-Primo and Richard J. Shavelson. Problems and issues in the use of Concept Maps in science assessment. *Journal of Research in Science Teaching, 33*(6): 569-600, 1996.

[78] Mehran Sahami, Mark Guzdial, Andrew McGettrick, and Steve Roach. Setting the Stage for Computing Curricula 2013: Computer Science – Report from the

ACM/IEEE-CS Joint Task Force. *Proceedings of the 42nd SIGCSE technical symposium on Computer science education,* , 2011.

[79] SBL. http://studiobasedlearning.org

[80] Scratch. http://scratch.mit.edu/

[81] Robert H. Sloan and Patrick Troy. CS 0.5: A Better Approach to Introductory Computer Science Majors. *Proceedings of the 39th SIGCSE technical symposium on Computer science education*, 271 - 275, 2008.

[82] Tracy E. Sutherland and Charles C. Bonwell (Eds.). Using active learning in college classes: A range of options for faculty. *New Directions for Teaching and Learning 67,* 1996.

[83] Allison Tew, Charles Fowler, and Mark Guzdial. Tracking an Innovation in Introductory CS Education from a Research University to a Two-Year College. *Proceedings of the 36th SIGCSE technical symposium on Computer science Education*, 416-420, 2005.

[84] Jay Vegso. Interest in CS as a Major Drops among Incoming Freshmen. *Computing Research News*, *17*(3):17, 2005.

[85] Jay Vegso. Interest in CS and CE as Majors Drops in 2005. *CRA Bulletin,* August 2006*,* Retrieved from *http://www.cra.org/wp/wp-trackback.php/75*

[86] Jay Vegso. Enrollment and Degree Production at US CS Departments drop further in 2006-2007. *Computing Research News,20*(2):4 , 2008.

[87] E. M. Wilcox, J. W. Atwood, M. M. Burnett, J. J. Cadiz, and C. R. Cook. Does continuous visual feedback aid debugging in direct-manipulation programming systems?. *Proceedings of the SIGCHI conference on Human factors in computing systems*, 258–265, 1997.

[88] Brenda C. Wilson and Sharon Shrock. Contributing to Success in an Introductory Computer Science Course: a Study of Twelve Factors. *Proceedings of the 32nd SIGCSE technical symposium on Computer Science Education,* 184-188, 2001.

[89] Ursula Wolz, John Maloney, and Sarah M. Pulimood. 'Scratch' Your Way to Introductory CS. *Proceedings of the 39th SIGCSE technical symposium on Computer science education,* 298-299, 2008.

[90] Michael Woodly and Samuel N. Kamin. Programming studio: a course for improving programming skills in undergraduates. *Proceedings of the 38th SIGCSE technical symposium on Computer science education,* 531 - 535, 2007.

[91] Stuart Zweben. Computing Degree and Enrollment Trends from the 2007-2008 CRA Taulbee Survey. *Computing Research Association,* Retrieved from, http://www.cra.org/taulbee/CRATaulbeeReport-StudentEnrollment-07-08.pdf

[92] Stuart Zweben. 2008-2009 Taulbee Survey: Undergraduate CS Enrolment Continues Rising; Doctoral Production Drops. *Computing Research Association,* Retrieved from, http://www.cra.org/uploads/documents/resources/taulbee/0809.pdf

[93] Stuart Zweben. 2009-2010 Taulbee Survey: Undergraduate CS Degree Production Rises; Doctoral Production Steady. *Computing Research Association,* Retrieved from, http://www.cra.org/uploads/documents/resources/taulbee/CRA_ Taulbee_2009-2010_Results.pdf

[94] Stuart Zweben. 2011-2012 Taulbee Survey: Computing Degree and Enrollment Trends. *Computing Research Association,* Retrieved from, http://cra.org/uploads/documents/resources/taulbee/CS_Degree_and_Enrollment_Trends_2010-11.pdf

[95] Stuart Zweben and Betsy Bizot. 2010-2011 Taulbee Survey: Continued Increase in Undergraduate CS Degree Production; Slight Rise in Doctoral Production. *Computing Research Association, 24*(3): 7-24, 2012.

TABLE OF APPENDIXES

Appendix J: CS116 – Exit Interview Questions

Lists interview questions along with directions for the interview at the end of the semester for student in CS116.

Appendix K: Transcripts of Entry Interview

A transcript of the conversation taken for a volunteer student in the entry interview.

Appendix L: Responses to the Exit Interview

A table of anonymous responses to the questions listed in the exit interview online form.

Appendix M: CS116 Students' Projects

A sampling of student projects submitted during the course at various stages.

APPENDIX A:

CS116 - Visual Programming Syllabus

Course Description

This course is a first exploration of the study of computing. It includes a broad survey of computer science including its history, ethics, programming languages, and computer architecture. It is an introduction for non-computer science majors in structured computer programming using non standard language. It offers modular programming techniques with emphasis on the creation of graphical user interfaces.

Prerequisite

No programming or computer science experience is required. Students should have sufficient facility with high-school mathematics to solve simple linear equations and to appreciate the use of mathematical notation and formalism.

Course Objectives

The student is expected to acquire knowledge of the development and use of computing technology, be able to make connections between computing technology and several other disciplines, be able to identify and explain significant implications of widespread use of computing technology, be able to think algorithmically rather than syntactically, and "become more proficient practitioners of computational problem solving"(studiobasedlearning.org).

By the end of this course, students will be able to:

1. **Identify** the historical background and technology that led to development and growth of computers. (Knowledge)
2. **Recognize** the basic physical building blocks that computers are constructed from and explain how they can be combined into computers. (Analysis)
3. **Understand and acquire** basic programming and algorithmic skills. (Knowledge and Comprehension)
4. **Use** language constructs in designing a smart phone application that may include text, sound, images and data structures. (Application)
5. **Compare and evaluate** the use of language constructs in different problems and situations. (Analysis)
6. **Identify** and **explain** machine languages, high-level languages, and basic issues involved in programming language translation. (Comprehension)
7. **Formulate** a reasonable theoretical model of computer programs and identify some of the problems that cannot be solved by such a model. (Synthesis)
8. **Identify** possible major trends in computer use and explain possible implications of such trends. (Evaluation)

9. **Solve and reflect on** algorithm and software design problems individually and collaboratively. (Application)
10. **Evaluate** alternate designs and **select** the best one using correctness, efficiency, and other software design issues as criterion. (Evaluation)
11. **Explain** selected solution using written and oral presentations. (Analysis)

(These objectives cover the important aspects students must take from this course, while using the levels of blooms taxonomy)

Course Content Outline and Format

The course will be taught using a studio based approach helping the students develop their own products and ideas. The lectures will discuss principles, concepts, theory, and examples as needed through the course. Open-lab assignments require students to work independently on individual or team projects. Some open-lab projects can be open-ended exercises. An outline of the course content follows.

1. History of computing.
2. Program translation: formal syntax, parsing and code generation.
3. Hardware and computer architecture.
4. Trends in computing.
5. Procedures, functions, and the use of arguments
6. Event handling
7. Control statements
8. Conditionals and random selection
9. Loops
10. Global vs. local variables
11. Strings and sting operations
12. Lists
13. Nesting of statements
14. Lexicographic vs. natural ordering
15. Runtime vs. compile time
16. Feasibility and project management
17. Use of manuals and documentation in creating projects
18. Peer programming
19. Client-server architecture
20. Copy rights and licenses

Course Assessment

Student learning outcomes will be assessed on the basis of assignments, exams, essays, and hands-on projects, which will be designed to evaluate the degree to which students meet the course objectives. Assignments, exams, and hands-on projects will require students to identify, evaluate, and use computing technology and explain its connections to other disciplines. Essays will require students to identify, explain, compare, and evaluate contemporary roles of computing technology in society, its connections to other disciplines, and implications of its use.

APPENDIX B:

Human Subjects Informed Consent Form for CS116

Study Title
Measuring the Impact of App Inventor for Android and Studio-Based Learning in an Introductory Computer Science Course for Non-Majors .

Study Purpose and Rationale
This study is designed to help us understand students' motivation and attitudes in introductory Computer Science courses for non-majors. We will also evaluate the efficacy of a new pedagogy and new tools for teaching introductory computer science. Findings from this study may lead to improved methods for teaching Computer Science, specifically to non-majors.

Inclusion/Exclusion Criteria
To be eligible to participate in this study, you must be at least 18 years of age and enrolled in CS116-001 in Fall 2009 at Ball State University.

Participation Procedure and Duration
Participants will complete the modified Motivated Strategies for Learning Questionnaires (MSLQ) at the beginning, middle, and end of the semester. Participants will also conduct entrance and exit interviews. Participants grant permission to use their MSLQ results, interview responses, course grades, GPA, and demographic data as part of this study.

Risks or Discomforts
This study has no foreseeable risks.

Benefits
This research may lead to a better understanding of non-CS majors' motivation and attitudes in CS courses, thereby leading to improved methods for teaching Computer Science to non-majors.

Confidentiality and Data Storage
The consent forms will be collected by Mr. Hamid Alkoot. The forms will be stored securely in the Computer Science Department office for the duration of this semester. Neither Dr. Gestwicki nor Ms. Ahmad will have access to these forms until the semester is over and your grades have been assigned. Prior to analysis of the collected data, each participant will be assigned an arbitrary identifier. This will be used to transcribe your MSLQ results, interview responses, course grades, academic profile (majors, minors, GPA, completed hours), and demographic data (gender, ethnicity, age). At this point, your name will be removed from the records, and no personally-identifying information will be disclosed as part of this study. Electronic records of grades and MSLQ results, indexed only by numeric identifier, will be kept on the instructor's workstation and the secure Computer Science departmental server. All other records will be destroyed.

Contact Information
For questions about your rights as a research subject, please contact Research Compliance, Office of Academic Research and Sponsored Programs, Ball State University, Muncie, IN 47306 (Telephone: 765-285-5070; Email: irb@bsu.edu).

For questions about this research study, contact Ms. Khuloud Ahmad, whose telephone number and email addresses are provided below.

About Participation
Your participation in this study is voluntary and you have the right to discontinue participation at any time without prejudice from the investigator.

Consent
I, _____, agree to participate in the research study titled "Measuring the Impact of App Inventor for Android and Studio-Based Learning in an Introductory Computer Science Course for Non-Majors ." I have had the study explained to me and my questions have been answered to my satisfaction. I have read the description of this project and give my consent to participate. I understand that I will receive a copy of this informed consent form to keep for future reference.

To the best of my knowledge, I meet the inclusion/exclusion criteria for this study.

_____ _____
Participant's Signature Date

Contact Information

Principal Investigators:

Paul V. Gestwicki, Ph.D.
Computer Science Department
Ball State University
Muncie, IN 47306
Telephone: 765-285-8668
Email: pvgestwicki@bsu.edu

Khuloud Ahmad
Computer Science Department
Ball State University
Muncie, IN 47306
Telephone: 765-212-1193
Email: knahmad@bsu.edu

APPENDIX C:

Human Subjects Informed Consent Form for CS110

Study Title
Measuring the Impact of App Inventor for Android and Studio-Based Learning in an Introductory Computer Science Course for Non-Majors.

Study Purpose and Rationale
This study is designed to help us understand students' motivation and attitudes in introductory Computer Science courses for non-majors. We will also evaluate the efficacy of a new pedagogy and new tools for teaching introductory computer science. Findings from this study may lead to improved methods for teaching Computer Science, specifically to non-majors.

Inclusion/Exclusion Criteria
To be eligible to participate in this study, you must be at least 18 years of age and enrolled in CS110-001 in Fall 2009 at Ball State University.

Participation Procedure and Duration
Participants will complete the modified Motivated Strategies for Learning Questionnaires (MSLQ) at the beginning and end of the semester. Participants grant permission to use their MSLQ results, grades, and demographic data as part of this study.

Risks or Discomforts
This study has no foreseeable risks.

Benefits
This research may lead to a better understanding of non-CS majors' motivation and attitudes in CS courses, thereby leading to improved methods for teaching Computer Science to non-majors.

Confidentiality and Data Storage
The consent forms will be collected by Khuloud Ahmad. The forms will be stored securely in the Computer Science Department office for the duration of this semester. Dr Sun will not have access to these forms.

Prior to analysis of the collected data, each participant will be assigned an arbitrary identifier. This will be used to transcribe your MSLQ results, grades, and demographic data (gender, ethnicity, age). At this point, your name will be removed from the records, and no personally-identifying information will be disclosed as part of this study.

Electronic records of grades and MSLQ results, indexed only by numeric identifier, will be kept on the instructor's workstation and the secure Computer Science departmental server. All other records will be destroyed.

Contact Information
For questions about your rights as a research subject, please contact Research Compliance, Office of Academic Research and Sponsored Programs, Ball State University, Muncie, IN 47306 (Telephone: 765-285-5070; Email: irb@bsu.edu).

For questions about this research study, contact Ms. Khuloud Ahmad, whose telephone number and email address are provided below.

About Participation
Your participation in this study is voluntary and you have the right to discontinue participation at any time without prejudice from the investigator.

Consent
I, _____, agree to participate in the research study titled "Measuring the Impact of App Inventor for Android and Studio-Based Learning in an Introductory Computer Science Course for Non-Majors ." I have had the study explained to me and my questions have been answered to my satisfaction. I have read the description of this project and give my consent to participate. I understand that I will receive a copy of this informed consent form to keep for future reference.

To the best of my knowledge, I meet the inclusion/exclusion criteria for this study.

_____ _____
Participant's Signature Date

Contact Information

Principal Investigators:

Paul V. Gestwicki, Ph.D.
Computer Science Department
Ball State University
Muncie, IN 47306
Telephone: 765-285-8668
Email: pvgestwicki@bsu.edu

Khuloud Ahmad
Computer Science Department
Ball State University
Muncie, IN 47306
Telephone: 765-212-1193
Email: knahmad@bsu.edu

APPENDIX D:

Recruitment Script for CS116

We invite you to participate in a computer science research study. The study is titled "Measuring the Impact of App Inventor for Android and Studio-Based Learning in an Introductory Computer Science Course for Non-Majors." This study will improve our understanding of students' motivation and attitudes in introductory Computer Science courses for non-majors and will also help in evaluating the efficacy of new pedagogy and software.

In this study, you will be answering three questionnaires, one at the beginning of the semester, one in the middle of the semester, and one at the end of the semester. The questionnaires ask you about your perspectives, expectations, study habits, learning skills, and motivation. We stress that these questionnaires are not tests and will not affect your grade in this course. During class time, everyone will be asked to answer the questionnaire; however, only the questionnaires of those who consent will be used in the study.

We also ask for volunteers to participate in interviews. In these interviews, you will be asked about your experiences, perspectives, and attitudes towards computer science in general and towards this course in particular. There will be a total of two interviews for each volunteer, one at the beginning of the semester and one at the end of the semester, and the interviews will not affect your grade in this course. The interviews will be conducted by a graduate student from the School of Music. The interviews will not be given to us – the researchers – until the grades for this course are posted, after the end of the semester. Those who complete the interviews will receive a $5 Barnes & Noble gift card for each interview.

Your participation is voluntary and not related in any way to your grade in this class. You may decide to participate now but you can withdraw from the study at any time without penalty. All your responses are strictly confidential and only members of the research team will see your individual responses. The researchers will not have access to your consent forms during the semester, and so we will not know who has volunteered to be in the study; participation will not affect your grade in any way.

You will receive a consent form shortly. If you agree to participate and are 18 years old or older, please sign the form. Also, if you choose to be interviewed, please email the interview coordinator at haalkoot@bsu.edu to set up an appointment.

We are happy to answer any questions you have, after which we will leave the room to protect your privacy. You may give your signed consent forms to Mr. Alkoot, who will ensure they are kept secure and private.

APPENDIX E:

Recruitment Script for CS110

I invite you to participate in a computer science research study. The study is titled "Measuring the Impact of App Inventor for Android and Studio-Based Learning in an Introductory Computer Science Course for Non-Majors." This study will improve our understanding of students' motivation and attitudes in introductory Computer Science courses for non-majors and will also help in evaluating the efficacy of new pedagogy and software.

In this study, you will be answering two questionnaires, one at the beginning of the semester and one at the end of the semester. The questionnaires ask you about your perspectives, expectations, study habits, learning skills, and motivation. We stress that these questionnaires are not tests and will not affect your grade in this course. During class time, everyone will be asked to answer the questionnaire; however, only the questionnaires of those who consent will be used in the study.

Your participation is voluntary and not related in any way to your grade in this class. You may decide to participate now but you can withdraw from the study at any time without penalty. All your responses are strictly confidential and only members of the research team will see your individual responses. The instructor will not have access to your consent forms during the semester, and so we will not know who has volunteered to be in the study; participation will not affect your grade in any way.

You will receive a consent form shortly. If you agree to participate and are 18 years old or older, please sign the form.

I am happy to answer any questions you have, after which Dr Sun will leave the room to protect your privacy.

APPENDIX F:

CS 116 — Visual Programming

Fall Semester, 2009

Course Entry Survey

We would like to ask for your participation in a study on attitudes towards computer science. The following survey will help us understand your attitudes regarding the CS-116 course. Your participation is voluntary and not related in any way to your grade in this class. You may decide to participate now but you can withdraw from the study at any time during the course of the semester with no penalty. All your responses are strictly confidential and only members of the research team will see your individual responses.

The attached questionnaire asks you about your study habits, your learning skills, your motivation for work in this course, and your future expectation and perspective of this course. THIS IS NOT A TEST. THERE ARE NO RIGHT OR WRONG ANSWERS TO THIS QUESTIONNAIRE. We want you to respond to the questionnaire as accurately as possible, reflecting your own attitudes and behaviors relative to this course.

Answer the questions of this survey relative to your expectations of what this course would be like or relative to your previous courses' experience, where appropriate.

We expect that this survey will require 20 – 30 minutes to complete. However, you may take as much time as you need.

Thanks in advance for your participation in this research!

Background Information

(All information collected will be kept strictly confidential and will not be associated with your name, per the informed consent agreement)

1. Name:_____

2. Declared major(s): _____

3. Declared minor(s): _____

4. How many hours per week do you study science subjects?
__ 0 __ 1-5 __ 6-10 __ 11-20 __ 21+

5. How difficult do you anticipate this course to be?
__ not difficult at all __ somewhat difficult __neutral __ difficult __very difficult

6. How likely is it that you will take another computer science course?
__ very unlikely __unlikely __ neutral __likely __ very likely

7. How important were the following for your decision to take this course/class.

	(1 = not at all important, 5 = very important)				
a. fulfills a course requirement	1	2	3	4	5
b. experience seemed interesting	1	2	3	4	5
c. is required	1	2	3	4	5
d. will be useful to me in school	1	2	3	4	5
e. will be useful to me in life	1	2	3	4	5
f. will help improve my academic skills	1	2	3	4	5
g. was recommended by a friend	1	2	3	4	5
h. was recommended by an advisor/professor	1	2	3	4	5
i. will improve career prospects	1	2	3	4	5
j. fit into my schedule	1	2	3	4	5

8. How confident are you in the following **(1 = not at all confident, 5 = very confident)**

a. engineering	1	2	3	4	5
b. math	1	2	3	4	5
c. reading	1	2	3	4	5
d. writing	1	2	3	4	5
e. science	1	2	3	4	5
f. computer science	1	2	3	4	5

9. How comfortable are you in the following **(1=not at all comfortable, 5=very comfortable)**

a. asking questions	1	2	3	4	5
b. collaborating with other students	1	2	3	4	5

MSLQ- Part A. Motivation

The following questions ask about your motivation for and attitudes about the class. **Remember there is no right or wrong answers; just please answer as accurately as possible.** Use the scale below to answer the questions. If you think the statement is very true of you, circle 7; if a statement is not at all true of you, circle 1. If the statement is more or less true of you, find a number between 1 and 7 that best describes you.

1	**2**	**3**	**4**	**5**	**6**	**7**
not at all true of me						**very true of me**

		not at all true of me						very true of me
1.	In a class like this, I prefer course material that really challenges me so I can learn new things.	1	2	3	4	5	6	7
2.	If I study in appropriate ways, then I will be able to learn the material in this course.	1	2	3	4	5	6	7
3.	I think I will be able to use what I learn in this course in other courses.	1	2	3	4	5	6	7
4.	I believe I will receive an excellent grade in this class.	1	2	3	4	5	6	7
5.	I'm certain I can understand the most difficult material that will be presented in the readings for this course.	1	2	3	4	5	6	7
6.	Getting a good grade in this class is the most satisfying thing for me right now.	1	2	3	4	5	6	7
7.	It is my own fault if I don't learn the material in this course.	1	2	3	4	5	6	7
8.	It is important for me to learn the course material in this class.	1	2	3	4	5	6	7
9.	The most important thing for me right now is improving my overall grade point average, so my main concern in this class is getting a good grade.	1	2	3	4	5	6	7

	not at all true of me						very true of me
10. I'm confident I can learn the basic concepts taught in this course.	1	2	3	4	5	6	7
11. If I can, I want to get better grades in this class then most of the other students.	1	2	3	4	5	6	7
12. I'm confident I can understand the most complex material that will be presented by the instructor in this course.	1	2	3	4	5	6	7
13. In a class like this, I prefer course material that arouses my curiosity, even if it is difficult to learn.	1	2	3	4	5	6	7
14. I am very interested in the content area of this course.	1	2	3	4	5	6	7
15. If I try hard enough, then I will understand the course material.	1	2	3	4	5	6	7
16. I'm confident I can do an excellent job on the assignments and tests in this course.	1	2	3	4	5	6	7
17. I expect to do well in this class.	1	2	3	4	5	6	7
18. The most satisfying thing for me in this course is trying to understand the content as thoroughly as possible.	1	2	3	4	5	6	7
19. I think the course material in this class is useful for me to learn.	1	2	3	4	5	6	7
20. When I have the opportunity in this class, I choose course assignments that I can learn from even if they don't guarantee a good grade.	1	2	3	4	5	6	7
21. If I don't understand the course material, it is because I didn't try hard enough.	1	2	3	4	5	6	7
22. I like the subject matter of this course.	1	2	3	4	5	6	7
23. Understanding the subject matter of this course is very important to me.	1	2	3	4	5	6	7

	not at all true of me						very true of me
24. I'm certain I can master the skills that will be taught in this class.	1	2	3	4	5	6	7
25. I want to do well in this class because it is important to show my ability to my family, friends, employer, or others.	1	2	3	4	5	6	7
26. Considering the difficulty of this course, the teacher, and my skills, I think I will do well in this class.	1	2	3	4	5	6	7

MSLQ - Part B. Learning Strategies

The following questions ask about your learning strategies and study skills. **Again, there is no right or wrong answers. Please answer the questions about how you study as accurately as possible.** Use the same scale to answer the remaining questions. If you think the statement is very true of you, circle 7; if a statement is not at all true of you, circle 1. If the statement is more or less true of you, find a number between 1 and 7 that best describes you.

1	**2**	**3**	**4**	**5**	**6**	**7**
not at all true of me						**very true of me**

	not at all true of me					very true of me
27. During class time I often miss important points because I'm thinking of other things.	1	2	3	4	5	6	7
28. When studying for such a course, I often try to explain the material to a classmate or friend.	1	2	3	4	5	6	7
29. I usually study in a place where I can concentrate on my course work.	1	2	3	4	5	6	7
30. When reading for such a course, I make up questions to help focus my reading.	1	2	3	4	5	6	7
31. I often feel so lazy or bored when I study for such a class that I quit before I finish what I planned to do.	1	2	3	4	5	6	7
32. I often find myself questioning things I hear or read in such a course to decide if I find them convincing.	1	2	3	4	5	6	7
33. Even if I have trouble learning the material in this class, I will try to do the work on my own, without help from anyone.	1	2	3	4	5	6	7
34. When I become confused about something I'm reading for this class, I will go back and try to figure it out.	1	2	3	4	5	6	7

	not at all true of me						very true of me
35. I will make good use of my study time for this course.	1	2	3	4	5	6	7
36. If course readings are difficult to understand, I will change the way I read the material.	1	2	3	4	5	6	7
37. I will try to work with other students from this class to complete the course assignments.	1	2	3	4	5	6	7
38. When a theory, interpretation, or conclusion is presented in class or in the readings, I try to decide if there is good supporting evidence.	1	2	3	4	5	6	7
39. I will work hard to do well in this class even if I don't like what we are doing.	1	2	3	4	5	6	7
40. When studying for such a course, I often set aside time to discuss course material with a group of students from the class.	1	2	3	4	5	6	7
41. I treat the course material as a starting point and try to develop my own ideas about it.	1	2	3	4	5	6	7
42. I find it hard to stick to a study schedule.	1	2	3	4	5	6	7
43. When I study for this course, I will pull together information from different sources, such as lectures, readings, and discussions.	1	2	3	4	5	6	7
44. Before I study new course material thoroughly, I often skim it to see how it is organized.	1	2	3	4	5	6	7
45. I ask myself questions to make sure I understand the material I have been studying in a class.	1	2	3	4	5	6	7
46. I try to change the way I study in order to fit the course requirements and the instructor's teaching style.	1	2	3	4	5	6	7
47. I often find that I have been reading for such a class but don't know what it is all about.	1	2	3	4	5	6	7

	not at all true of me						very true of me
48. I ask the instructor to clarify concepts I don't understand well.	1	2	3	4	5	6	7
49. When course work is difficult, I either give up or only study the easy parts.	1	2	3	4	5	6	7
50. I try to think through a topic and decide what I am supposed to learn from it rather than just reading it over when studying for such a course.	1	2	3	4	5	6	7
51. I will try to relate ideas in this subject to those in other courses whenever possible.	1	2	3	4	5	6	7
52. When reading for this class, I will try to relate the material to what I already know.	1	2	3	4	5	6	7
53. I have a regular place set aside for studying.	1	2	3	4	5	6	7
54. I will try to play around with ideas of my own related to what I will be learning in this course.	1	2	3	4	5	6	7
55. When I study for this course, I will write brief summaries of the main ideas from the readings and my class notes.	1	2	3	4	5	6	7
56. When I can't understand the material in this course, I will ask another student in this class for help.	1	2	3	4	5	6	7
57. I will try to understand the material in this class by making connections between the readings and the concepts from the lectures.	1	2	3	4	5	6	7
58. I will make sure that I keep up with the weekly readings and assignments for this course.	1	2	3	4	5	6	7
59. Whenever I read or hear an assertion or conclusion in such a class, I think about possible alternatives.	1	2	3	4	5	6	7
60. I will attend this class regularly.	1	2	3	4	5	6	7

	not at all true of me						very true of me
61. Even when course materials are dull and uninteresting, I manage to keep working until I finish.	1	2	3	4	5	6	7
62. I will try to identify students in this class whom I can ask for help if necessary.	1	2	3	4	5	6	7
63. When studying for such a course I try to determine which concepts I don't understand well.	1	2	3	4	5	6	7
64. I often find that I don't spend very much time on such as course because of other activities.	1	2	3	4	5	6	7
65. When I study for such a class, I set goals for myself in order to direct my activities in each study period.	1	2	3	4	5	6	7
66. If I get confused taking notes in class, I make sure I sort it out afterwards.	1	2	3	4	5	6	7
67. I rarely find time to review my notes of readings before an exam.	1	2	3	4	5	6	7
68. I try to apply ideas from course readings in other class activities such as lecture and discussion.	1	2	3	4	5	6	7

APPENDIX G:

CS 116 — Visual Programming

Fall Semester, 2009

Course Mid Survey

The following survey will help us understand your attitudes regarding the CS-116 course. Your participation is voluntary and not related in any way to your grade in this class. You may decide to participate now but you can withdraw from the study at any time during the course of the semester with no penalty. All your responses are strictly confidential and only members of the research team will see your individual responses.

The attached questionnaire asks you about your study habits, your learning skills, and your motivation for work in this course. THIS IS NOT A TEST. THERE ARE NO RIGHT OR WRONG ANSWERS TO THIS QUESTIONNAIRE. We want you to respond to the questionnaire as accurately as possible, reflecting your own attitudes and behaviors in this course.

We anticipate that this survey will require 20 – 30 minutes to complete. However, you may take as much time as you need.

Thanks in advance for your participation in this research!

Background Information

(All information collected will be kept strictly confidential and will not be associated with your name, per the informed consent agreement)

1. Name:_____

2. How many hours per week do you prepare for this course?
__ 0 __ 1-5 __ 6-10 __ 11-20 __ 21+

3. How difficult do you find this course?
__ not difficult at all __ somewhat difficult __neutral __ difficult __very difficult

4. How likely is it that you will take another computer science course?
__ very unlikely __unlikely __ neutral __likely __ very likely

5. How likely is it that you will recommend this course to other students?
__ very unlikely __unlikely __ neutral __likely __ very likely

6. How intimidated do you feel in the class? **(1 = very intimidated, 5 = not at all intimidated)**
1 2 3 4 5

7. How comfortable are you in the following **(1=not at all comfortable, 5=very comfortable)**
a. asking questions 1 2 3 4 5
b. collaborating with other students 1 2 3 4 5

MSLQ- Part A. Motivation

The following questions ask about your motivation for and attitudes about the class. **Remember there is no right or wrong answers; just please answer as accurately as possible.** Use the scale below to answer the questions. If you think the statement is very true of you, circle 7; if a statement is not at all true of you, circle 1. If the statement is more or less true of you, find a number between 1 and 7 that best describes you.

1 2 3 4 5 6 7
not at all true of me **very true of me**

		not at all true of me						very true of me
1.	In a class like this, I prefer course material that really challenges me so I can learn new things.	1	2	3	4	5	6	7
2.	If I study in appropriate ways, then I will be able to learn the material in this course.	1	2	3	4	5	6	7
3.	I think I will be able to use what I learn in this course in other courses.	1	2	3	4	5	6	7
4.	I believe I will receive an excellent grade in this class.	1	2	3	4	5	6	7
5.	I'm certain I can understand the most difficult material presented in the readings for this course.	1	2	3	4	5	6	7
6.	Getting a good grade in this class is the most satisfying thing for me right now.	1	2	3	4	5	6	7
7.	It is my own fault if I don't learn the material in this course.	1	2	3	4	5	6	7
8.	It is important for me to learn the course material in this class.	1	2	3	4	5	6	7
9.	The most important thing for me right now is improving my overall grade point average, so my main concern in this class is getting a good grade.	1	2	3	4	5	6	7
10.	I'm confident I can learn the basic concepts taught in this course.	1	2	3	4	5	6	7

	not at all true of me						very true of me
11. If I can, I want to get better grades in this class then most of the other students.	1	2	3	4	5	6	7
12. I'm confident I can understand the most complex material presented by the instructor in this course.	1	2	3	4	5	6	7
13. In a class like this, I prefer course material that arouses my curiosity, even if it is difficult to learn.	1	2	3	4	5	6	7
14. I am very interested in the content area of this course.	1	2	3	4	5	6	7
15. If I try hard enough, then I will understand the course material.	1	2	3	4	5	6	7
16. I'm confident I can do an excellent job on the assignments and tests in this course.	1	2	3	4	5	6	7
17. I expect to do well in this class.	1	2	3	4	5	6	7
18. The most satisfying thing for me in this course is trying to understand the content as thoroughly as possible.	1	2	3	4	5	6	7
19. I think the course material in this class is useful for me to learn.	1	2	3	4	5	6	7
20. When I have the opportunity in this class, I choose course assignments that I can learn from even if they don't guarantee a good grade.	1	2	3	4	5	6	7
21. If I don't understand the course material, it is because I didn't try hard enough.	1	2	3	4	5	6	7
22. I like the subject matter of this course.	1	2	3	4	5	6	7
23. Understanding the subject matter of this course is very important to me.	1	2	3	4	5	6	7

	not at all true of me						very true of me
24. I'm certain I can master the skills being taught in this class.	1	2	3	4	5	6	7
25. I want to do well in this class because it is important to show my ability to my family, friends, employer, or others.	1	2	3	4	5	6	7
26. Considering the difficulty of this course, the teacher, and my skills, I think I will do well in this class.	1	2	3	4	5	6	7

MSLQ - Part B. Learning Strategies

The following questions ask about your learning strategies and study skills. **Again, there is no right or wrong answers. Please answer the questions about how you study as accurately as possible.** Use the same scale to answer the remaining questions. If you think the statement is very true of you, circle 7; if a statement is not at all true of you, circle 1. If the statement is more or less true of you, find a number between 1 and 7 that best describes you.

1	**2**	**3**	**4**	**5**	**6**	**7**
not at all true of me						**very true of me**

	not at all true of me					very true of me
27. During class time I often miss important points because I'm thinking of other things.	1	2	3	4	5	6	7
28. When studying for this course, I often try to explain the material to a classmate or friend.	1	2	3	4	5	6	7
29. I usually study in a place where I can concentrate on my course work.	1	2	3	4	5	6	7
30. When reading for this course, I make up questions to help focus my reading.	1	2	3	4	5	6	7
31. I often feel so lazy or bored when I study for this class that I quit before I finish what I planned to do.	1	2	3	4	5	6	7
32. I often find myself questioning things I hear or read in this course to decide if I find them convincing.	1	2	3	4	5	6	7
33. Even if I have trouble learning the material in this class, I try to do the work on my own, without help from anyone.	1	2	3	4	5	6	7
34. When I become confused about something I'm reading for this class, I go back and try to figure it out.	1	2	3	4	5	6	7
35. I make good use of my study time for this course.	1	2	3	4	5	6	7

	not at all true of me						very true of me
36. If course readings are difficult to understand, I change the way I read the material.	1	2	3	4	5	6	7
37. I try to work with other students from this class to complete the course assignments.	1	2	3	4	5	6	7
38. When a theory, interpretation, or conclusion is presented in class or in the readings, I try to decide if there is good supporting evidence.	1	2	3	4	5	6	7
39. I work hard to do well in this class even if I don't like what we are doing.	1	2	3	4	5	6	7
40. When studying for this course, I often set aside time to discuss course material with a group of students from the class.	1	2	3	4	5	6	7
41. I treat the course material as a starting point and try to develop my own ideas about it.	1	2	3	4	5	6	7
42. I find it hard to stick to a study schedule.	1	2	3	4	5	6	7
43. When I study for this course, I pull together information from different sources, such as lectures, readings, and discussions.	1	2	3	4	5	6	7
44. Before I study new course material thoroughly, I often skim it to see how it is organized.	1	2	3	4	5	6	7
45. I ask myself questions to make sure I understand the material I have been studying in this class.	1	2	3	4	5	6	7
46. I try to change the way I study in order to fit the course requirements and the instructor's teaching style.	1	2	3	4	5	6	7
47. I often find that I have been reading for this class but don't know what it was all about.	1	2	3	4	5	6	7
48. I ask the instructor to clarify concepts I don't understand well.	1	2	3	4	5	6	7

	not at all true of me						very true of me
49. When course work is difficult, I either give up or only study the easy parts.	1	2	3	4	5	6	7
50. I try to think through a topic and decide what I am supposed to learn from it rather than just reading it over when studying for this course.	1	2	3	4	5	6	7
51. I try to relate ideas in this subject to those in other courses whenever possible.	1	2	3	4	5	6	7
52. When reading for this class, I try to relate the material to what I already know.	1	2	3	4	5	6	7
53. I have a regular place set aside for studying.	1	2	3	4	5	6	7
54. I try to play around with ideas of my own related to what I am learning in this course.	1	2	3	4	5	6	7
55. When I study for this course, I write brief summaries of the main ideas from the readings and my class notes.	1	2	3	4	5	6	7
56. When I can't understand the material in this course, I ask another student in this class for help.	1	2	3	4	5	6	7
57. I try to understand the material in this class by making connections between the readings and the concepts from the lectures.	1	2	3	4	5	6	7
58. I make sure that I keep up with the weekly readings and assignments for this course.	1	2	3	4	5	6	7
59. Whenever I read or hear an assertion or conclusion in this class, I think about possible alternatives.	1	2	3	4	5	6	7
60. I attend this class regularly.	1	2	3	4	5	6	7
61. Even when course materials are dull and uninteresting, I manage to keep working until I finish.	1	2	3	4	5	6	7
62. I try to identify students in this class whom I can ask for help if necessary.	1	2	3	4	5	6	7

	not at all true of me						very true of me
63. When studying for this course I try to determine which concepts I don't understand well.	1	2	3	4	5	6	7
64. I often find that I don't spend very much time on this course because of other activities.	1	2	3	4	5	6	7
65. When I study for this class, I set goals for myself in order to direct my activities in each study period.	1	2	3	4	5	6	7
66. If I get confused taking notes in class, I make sure I sort it out afterwards.	1	2	3	4	5	6	7
67. I rarely find time to review my notes of readings before an exam.	1	2	3	4	5	6	7
68. I try to apply ideas from course readings in other class activities such as lecture and discussion.	1	2	3	4	5	6	7

APPENDIX H:

CS 116 — Visual Programming
Fall Semester, 2009

Course Exit Survey

The following survey will help us understand your attitudes regarding the CS-116 course. Your participation is voluntary and not related in any way to your grade in this class. You may decide to participate now but you can withdraw from the study at any time during the course of the semester with no penalty. All your responses are strictly confidential and only members of the research team will see your individual responses.

The attached questionnaire asks you about your study habits, your learning skills, and your motivation for work in this course. THIS IS NOT A TEST. THERE ARE NO RIGHT OR WRONG ANSWERS TO THIS QUESTIONNAIRE. We want you to respond to the questionnaire as accurately as possible, reflecting your own attitudes and behaviors in this course.

We anticipate that this survey will require 20 – 30 minutes to complete. However, you may take as much time as you need.

Thanks in advance for your participation in this research!

Background Information

(All information collected will be kept strictly confidential and will not be associated with your name, per the informed consent agreement)

1. Name:_____

2. How many hours per week do you prepare for this course?
__ 0 __ 1-5 __ 6-10 __ 11-20 __ 21+

3. How difficult do you find this course?
__ not difficult at all __ somewhat difficult __neutral __ difficult __very difficult

4. How likely is it that you will take another computer science course?
__ very unlikely __unlikely __ neutral __likely __ very likely

5. How likely is it that you will recommend this course to other students?
__ very unlikely __unlikely __ neutral __likely __ very likely

6. How intimidated do you feel in the class? **(1 = very intimidated, 5 = not at all intimidated)**
1 2 3 4 5

7. How comfortable are you in the following **(1=not at all comfortable, 5=very comfortable)**
a. asking questions 1 2 3 4 5
b. collaborating with other students 1 2 3 4 5

MSLQ- Part A. Motivation

The following questions ask about your motivation for and attitudes about the class. **Remember there is no right or wrong answers; just please answer as accurately as possible.** Use the scale below to answer the questions. If you think the statement is very true of you, circle 7; if a statement is not at all true of you, circle 1. If the statement is more or less true of you, find a number between 1 and 7 that best describes you.

1 2 3 4 5 6 7
not at all true of me **very true of me**

		not at all true of me						very true of me
1.	In a class like this, I prefer course material that really challenges me so I can learn new things.	1	2	3	4	5	6	7
2.	If I study in appropriate ways, then I will be able to learn the material in this course.	1	2	3	4	5	6	7
3.	I think I will be able to use what I learn in this course in other courses.	1	2	3	4	5	6	7
4.	I believe I will receive an excellent grade in this class.	1	2	3	4	5	6	7
5.	I'm certain I can understand the most difficult material presented in the readings for this course.	1	2	3	4	5	6	7
6.	Getting a good grade in this class is the most satisfying thing for me right now.	1	2	3	4	5	6	7
7.	It is my own fault if I don't learn the material in this course.	1	2	3	4	5	6	7
8.	It is important for me to learn the course material in this class.	1	2	3	4	5	6	7
9.	The most important thing for me right now is improving my overall grade point average, so my main concern in this class is getting a good grade.	1	2	3	4	5	6	7
10.	I'm confident I can learn the basic concepts taught in this course.	1	2	3	4	5	6	7

	not at all true of me						very true of me
11. If I can, I want to get better grades in this class then most of the other students.	1	2	3	4	5	6	7
12. I'm confident I can understand the most complex material presented by the instructor in this course.	1	2	3	4	5	6	7
13. In a class like this, I prefer course material that arouses my curiosity, even if it is difficult to learn.	1	2	3	4	5	6	7
14. I am very interested in the content area of this course.	1	2	3	4	5	6	7
15. If I try hard enough, then I will understand the course material.	1	2	3	4	5	6	7
16. I'm confident I can do an excellent job on the assignments and tests in this course.	1	2	3	4	5	6	7
17. I expect to do well in this class.	1	2	3	4	5	6	7
18. The most satisfying thing for me in this course is trying to understand the content as thoroughly as possible.	1	2	3	4	5	6	7
19. I think the course material in this class is useful for me to learn.	1	2	3	4	5	6	7
20. When I have the opportunity in this class, I choose course assignments that I can learn from even if they don't guarantee a good grade.	1	2	3	4	5	6	7
21. If I don't understand the course material, it is because I didn't try hard enough.	1	2	3	4	5	6	7
22. I like the subject matter of this course.	1	2	3	4	5	6	7
23. Understanding the subject matter of this course is very important to me.	1	2	3	4	5	6	7

	not at all true of me						very true of me
24. I'm certain I can master the skills being taught in this class.	1	2	3	4	5	6	7
25. I want to do well in this class because it is important to show my ability to my family, friends, employer, or others.	1	2	3	4	5	6	7
26. Considering the difficulty of this course, the teacher, and my skills, I think I will do well in this class.	1	2	3	4	5	6	7

MSLQ - Part B. Learning Strategies

The following questions ask about your learning strategies and study skills. **Again, there is no right or wrong answers. Please answer the questions about how you study as accurately as possible.** Use the same scale to answer the remaining questions. If you think the statement is very true of you, circle 7; if a statement is not at all true of you, circle 1. If the statement is more or less true of you, find a number between 1 and 7 that best describes you.

1	**2**	**3**	**4**	**5**	**6**	**7**
not at all true of me						**very true of me**

	not at all true of me					very true of me
27. During class time I often miss important points because I'm thinking of other things.	1	2	3	4	5	6	7
28. When studying for this course, I often try to explain the material to a classmate or friend.	1	2	3	4	5	6	7
29. I usually study in a place where I can concentrate on my course work.	1	2	3	4	5	6	7
30. When reading for this course, I make up questions to help focus my reading.	1	2	3	4	5	6	7
31. I often feel so lazy or bored when I study for this class that I quit before I finish what I planned to do.	1	2	3	4	5	6	7
32. I often find myself questioning things I hear or read in this course to decide if I find them convincing.	1	2	3	4	5	6	7
33. Even if I have trouble learning the material in this class, I try to do the work on my own, without help from anyone.	1	2	3	4	5	6	7
34. When I become confused about something I'm reading for this class, I go back and try to figure it out.	1	2	3	4	5	6	7
35. I make good use of my study time for this course.	1	2	3	4	5	6	7

	not at all true of me						very true of me
36. If course readings are difficult to understand, I change the way I read the material.	1	2	3	4	5	6	7
37. I try to work with other students from this class to complete the course assignments.	1	2	3	4	5	6	7
38. When a theory, interpretation, or conclusion is presented in class or in the readings, I try to decide if there is good supporting evidence.	1	2	3	4	5	6	7
39. I work hard to do well in this class even if I don't like what we are doing.	1	2	3	4	5	6	7
40. When studying for this course, I often set aside time to discuss course material with a group of students from the class.	1	2	3	4	5	6	7
41. I treat the course material as a starting point and try to develop my own ideas about it.	1	2	3	4	5	6	7
42. I find it hard to stick to a study schedule.	1	2	3	4	5	6	7
43. When I study for this course, I pull together information from different sources, such as lectures, readings, and discussions.	1	2	3	4	5	6	7
44. Before I study new course material thoroughly, I often skim it to see how it is organized.	1	2	3	4	5	6	7
45. I ask myself questions to make sure I understand the material I have been studying in this class.	1	2	3	4	5	6	7
46. I try to change the way I study in order to fit the course requirements and the instructor's teaching style.	1	2	3	4	5	6	7
47. I often find that I have been reading for this class but don't know what it was all about.	1	2	3	4	5	6	7
48. I ask the instructor to clarify concepts I don't understand well.	1	2	3	4	5	6	7
49. When course work is difficult, I either give up or only study the easy parts.	1	2	3	4	5	6	7

	not at all true of me						very true of me
50. I try to think through a topic and decide what I am supposed to learn from it rather than just reading it over when studying for this course.	1	2	3	4	5	6	7
51. I try to relate ideas in this subject to those in other courses whenever possible.	1	2	3	4	5	6	7
52. When reading for this class, I try to relate the material to what I already know.	1	2	3	4	5	6	7
53. I have a regular place set aside for studying.	1	2	3	4	5	6	7
54. I try to play around with ideas of my own related to what I am learning in this course.	1	2	3	4	5	6	7
55. When I study for this course, I write brief summaries of the main ideas from the readings and my class notes.	1	2	3	4	5	6	7
56. When I can't understand the material in this course, I ask another student in this class for help.	1	2	3	4	5	6	7
57. I try to understand the material in this class by making connections between the readings and the concepts from the lectures.	1	2	3	4	5	6	7
58. I make sure that I keep up with the weekly readings and assignments for this course.	1	2	3	4	5	6	7
59. Whenever I read or hear an assertion or conclusion in this class, I think about possible alternatives.	1	2	3	4	5	6	7
60. I attend this class regularly.	1	2	3	4	5	6	7
61. Even when course materials are dull and uninteresting, I manage to keep working until I finish.	1	2	3	4	5	6	7
62. I try to identify students in this class whom I can ask for help if necessary.	1	2	3	4	5	6	7
63. When studying for this course I try to determine which concepts I don't understand well.	1	2	3	4	5	6	7

	not at all true of me						very true of me
64. I often find that I don't spend very much time on this course because of other activities.	1	2	3	4	5	6	7
65. When I study for this class, I set goals for myself in order to direct my activities in each study period.	1	2	3	4	5	6	7
66. If I get confused taking notes in class, I make sure I sort it out afterwards.	1	2	3	4	5	6	7
67. I rarely find time to review my notes of readings before an exam.	1	2	3	4	5	6	7
68. I try to apply ideas from course readings in other class activities such as lecture and discussion.	1	2	3	4	5	6	7

APPENDIX I:

CS 116 – Visual Programming, Fall 2009

Entry Interview Questions

Interviewer's Directions:

1- At the start of the interview, read the following passage:

"The issue of interest here is learning more about your experiences with CS 116. Your answers will help better understand the impact of the course, and will help improve the course. Nothing that you say in this interview will impact your course grade in any way. The instructors will not know your identity and the interviews will be kept sealed until after the grades are submitted. Your honesty is highly valued, and I would like to encourage you to speak your mind openly. This interview should last roughly 30 minutes, and it will be recorded. Before we begin, do you have any questions?"

2- After answering any questions the interviewee had, engage him with some general chatting as an icebreaker and to set the student at ease, for example, "how are you?"

3- Start with the interview questions.

Interview questions:

1. Why did you enroll in this course?
2. What are your expectations for this course?
3. How is Computer Science relevant to your major?
4. How is Computer Science relevant to your personal life?
5. From your perspective, what do computer scientists do?
6. What skills do you think are important for computer scientists?
7. Have you ever considered majoring in Computer Science?
8. Have you had any prior experience with computer programming, and how did you feel about it?
9. How comfortable are you working with others to solve problems?
10. Have you done any peer evaluations in your previous courses? If so, how were they useful or not useful?
11. How would you compare peer evaluation to instructor evaluation? Do you prefer one over the other?

APPENDIX J:

CS 116 – Visual Programming, Fall 2009

Exit Interview Questions

Interviewer's Directions:

1- At the start of the interview, read the following passage:

"The issue of interest here is learning more about your experiences with CS 116. Your answers will help better understand the impact of the course, and will help improve the course. Nothing that you say in this interview will impact your course grade in any way. The instructors will not know your identity and the interviews will be kept sealed until after the grades are submitted. Your honesty is highly valued, and I would like to encourage you to speak your mind openly. This interview should last roughly 30 minutes, and it will be recorded. Before we begin, do you have any questions?"

2- After answering any questions the interviewee had, engage him with some general chatting as an icebreaker and to set the student at ease, for example, "how are you?"

3- Start with the interview questions.

Interview questions:

1. Did this course meet your expectations? Please explain.
2. What aspect of the course was most interesting, and why?
3. What did you think about the programming projects and studios?
4. Were the projects and studios beneficial to your learning?
5. How comfortable are you working with others to solve problems?
6. How would you compare peer evaluation to instructor evaluation? Do you prefer one over the other?
7. Is there anything that you would change about the design of the course? Please explain.
8. How is Computer Science relevant to your major?
9. How is Computer Science relevant to your personal life?

10. What skills do you think are important for computer scientists?
11. Did this course change the way you think about Computer Science?
12. What did you think about App Inventor for Android?
13. What was your favorite aspect of using App Inventor for Android?
14. What was your least favorite aspect of using App Inventor for Android?

APPENDIX K:

Transcripts of Entry Interview

I = interviewer
S = student

I: The first question is why did you enroll in this course?
S: cause I am taking a computer , you know, computer major design.

I: so you are majoring in computers?
S: yeh.

I: ok. So is it a requirement or elective for you?
S: I think it is an elective for me.

I: so you chose it.
S: yeh yeh.

I: and so what were your expectations of this course?
S: to learn a lot about computers I did not know, like, before.

I: like what, what did you expect to learn?
S: like you know the program, the computer and the software and stuff like that.

I: so learning about different software maybe? Know how to program?
S: yeh……yeh.

I: ok. How is computer science relevant to your major? I think we answered that it's an elective you said?
S: yeh.

I: is it an open elective or is it an elective of your program of study?
S: it is an elective in my program.

I: ok. And how is computer science relevant to your personal life?
S: how…….. what……?

I: how is computer science relevant to your personal life?

S: am cause I got in computers a lot, so kind of like understand like how to like try to work computers and stuff.

I: ok
S: so how computers work and stuff.

I: so how do you use it, daily?
S: yeh yeh I use it daily.

I: to work or to gaming or for what?
S: I use it every day for a lot of different things.

I: ok. So word processing and stuff like that?
S: yeh like internet and gaming.

I: but nothing more deep like programming or something like that?
S: no ..ah no.

I: ok. So from your perspective, what do computer scientists do?
S: aha probably ……

I: what do you think they do?
S: I think they probably like study computers and trying to figure out different ways to make computers better. You know you make a way you can do a lot of easier thing on computers.

I: ok. Do you think they are involved in software or hardware?
S: probably both… probably both.

I: ok. So what skills do you think are important for computer scientists?
S: amm probably amm skills, probably like amm being able to like have patient with the computer like you know if they are going slow so you know to not get made about it so you know just you know have patient.

I: ok. So is there anything else that they need to know to be able to do their job?
S: amm probably just like patient I am not really sure exactly what they …..

I: yeh sure. This interview intended, its early in the semester so that we have an early perspective of what you know and then at the last of the semester so we see if something is changed or something different. So it's ok if you just don't know.

S: all right.

I: but I was asking about you expectations of what skills a computer scientist may need to be able to do his job, like a doctor need certain criteria to be a doctor.
S: yeh.
I: so do you thing computer scientists need a certain criteria or just anybody could do a computer scientist's job?
S: amm I don't amm quite, anybody can do it, like to be a sience person like certain thing you need.

I: ok. Have you ever consider majoring in computer science?
S: well if in computer science like you doing hardware then yeh that is what I want to do.

I: could you elaborate on that? So you would consider it if it was dealing with certain things with computers not other things is that right?
S: yeh.

I: so what would make you consider this major or not?
S: amm like amm if I want like consider it a main like I mean what kind of software or not so I would prefer like to build computer like rather than to deal with software.

I: ok so like to build the computer and put the parts and everything and to put it together and see how it works?
S: yeh yeh.

I: more than software and programming then?
S: yeh yeh.

I: ok. Have you had any prior experience with computer programming?
S: I had a class I already that talk about programming and stuff.

I: ok. Do you remember what languages you took or anything?
S: amm I remember something, I think I get to know some things like Microsoft word and stuff like that, you know like dealing with word processing and stuff like that.

I: ok. So it was more about using the software not programming the software.
S: yeh yeh.

I: ok. And how did you feel about it?

S: amm it was kind of like confusing at times like you know and there were time that it was easy and sometimes it was hard.

I: ok. Do you remember what was confusing about it? Was it the teacher or the material or what?
S: oh it was the material. Like sometimes it was hard to follow like it was hard to follow and listen at the same time.
I: ok. So was it a group session or?
S: yeh it was a group.

I: ok. How comfortable are you working with others to solve problems?
S: amm no like I amm like, I can work with other people but I prefer to work by myself.

I: you prefer to work alone?
S: yeh.

I: so you feel you can achieve more working alone?
S: yeh yeh.

I: ok. Don't you feel that when working in groups you benefit of having more different ideas?
S: ah no I feel there is an advantage of working along like amm you can concentrate more you know what I mean? Like when I was in high school a lot of people were like interacting and like not paying attention they just… so I am more like have As and Bs in high school so like you know being alone.

I: yeh I understand. Ok. Have you done any peer evaluations in your previous courses?
S: ah no haven't done that.

I: do you know what is peer evaluation?
S: ah … not sure exactly.

I: peer evaluation is done in the class where the teacher allow students to evaluate other students' work and maybe make suggestions of their work. So it's not actually grading but more of exchanging ideas and critic.
S: yeh when I think about it I think I've done that in previous class like you write a paper review and like like your paper gets feedback of what you done.

I: ok. So did you find this peer review useful or not?

S: I think it was very useful cause you get some notes and you have to figure out if these notes are right or not cause sometimes they can lead you in the wrong direction.

I: yeh. So how do you compare peer evaluation with instructor evaluation?
S: I think the instructor's evaluations they they help you a lot better because they know what they are grading on.

I: yeh.
S: and the students you know they show their opinions and stuff.
I: so you feel that it is more informed grading and opinions from the instructor?
S: yeh.

I: than maybe some feedback from other students?
S: yeh yeh.

I: so you prefer the instructor doing the review than the students?
S: yeh.

I: ok. I think we are done with our questions with you. So we should have another interview at the end of the semester, maybe in the last two weeks of the semester.
S: oh cool. ahh you will email me or how would that works?

I: yeh you can email me maybe at the beginning or the middle of November. And I will email you to remind you. I know you will forget because.
S: yeh yeh.

I: so just keep in mind we will have another one like this at the end of the semester.
S: ok.

I: this is the card and if you have any other questions?
S: ah no that's it.

I: ok. Thank you very much.
S: ah thanks.

APPENDIX L:

Responses to the Exit Interview

Questions \ Respondents	Respondent_1	Respondent_2	Respondent_3	Respondent_4	Respondent_5	Respondent_6	Respondent_7
1. Describe any programming experiences you had before this course. Please include course numbers if appropriate	I had no programming experience before this course.	CS 110 was the start of javascript for myself but prior to that I was very familiar with HTML. I'm not certain how much HTML counts towards programming but I suppose it has some relevant tie.	None	i didnt have a programming course before i took this course.	I have had no experiences with programming before this class.	I haven't taken any other courses.	No Prior programming experience.
2. Did this course meet your expectations?	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Please explain.	I expected to get a basic about what computer programming was about and this course was adequate.	It went above and beyond my expectations. This is the first time at Ball State that I have ever had the so called "immersive learning" experience.	I didnt really have any expectations when entering the class.	i learned more than i knew when i camt to this class on the first day. This was my expectations just to learn somthing i didnt know before i started this cours. Now i feel like i understand more than i did on the first day.	Yes, I learned a lot because I entered this class to little to none programming experience and my skills with the programming grew a great deal over the course of this semester.	The course exceeded my expectations for the class.	I expected this course to be kind of an introduction to the basic concepts of code. I thought that through the use of the mobile applications, and writing code for these, that i gained some basic knowledge and experience to write some on my own.
3. What aspect of the course was most interesting, and why?	The ability to work with google and their software because they are are top of the line franchise and they are asking us for help.	Everything. I was incredibly excited to get the most out of this course because I had been dying to try out the Android OS.	I just found it fun to be able to create an application where you saw the final product.	The most interesting apect of this course was trying to come up with a project that met the expectations and works too. I had some problems trying to pick a project that did both. Towards the end of class i started understanding the material much better.	The most interesting aspect of this course was the program that we used to build the applications. I just liked how it was a visual building block application.	I liked the fact that we learned the basics of what makes applications work and how to design our own. Getting to work with the Android phones was an added aesthetic to the class as well.	I think the most interesting aspect of this course was the actual building and constructing the code in the blocks editor. Sure the design view was interesting, but i liked actually getting into the physical aspect of the code.
4. What did you think about the programming projects and studios?	The projects were all worth the effort and everything made you get outside your comfort zone.	I'm not entirely sure what is meant by studios. Otherwise I can assure you that the projects were a lot of fun. I love being creative whenever I can so this was another awesome outlet for that creativity.	I found it fun, but I wouldnt want to do it for the rest of my life.	I thought sometimes it was challenging. I also thought that i did my very best to understand what was being taught to me. I think that the projects went better than i expected it would go on the first day.	THe projects were very fun and I liked that the students were in full control over the projects that they were working on.	I really liked how we were able to work in groups to do our projects and the groups changed a few times so we could work with new people and get to know them better.	I believe that the projects that we did in the class were exciting and i really liked how we could work in groups for most of them.s

5. Were the projects and studios beneficial to your learning?	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Please explain.	I learned a lot about programming how different aspects work cohesively.	Anything in the course would be beneficial because it was all new and now I can create so many more useful things, even make things for use in other courses.	well sort of, this class taught me to think more logically but that would be about it, since I dont plan to do any programming in the future.	They were because they allowed us to meet people and expand my learning. I was more willing to reach out to people for help. This was because i was able to do projects on my own.	I learned a lot because with each project came a knew problem or problems that had to be solved.	I really enjoyed doing projects and working in the studios.	I love studio based learning and i learn more in these kinds of environments.
6. How comfortable are you working with others to solve problems?	I have no problem working with other people to solve problems. The more people, the easier it will be to find the solution.	For the most part I do not like to work with other people unless I've known them a long time and know how to deal with them. But otherwise I have never liked working with others because it seems that I am the only one that takes initiative.	pretty comfortable.	I'm not extrememly comfortable but i am comfortable to get the task done with others. I will work with people to meet the expectations if required or advised. So i am little comfortable.	Working with partners was one of my favorite aspects of the course because we would help and teach each other.	This class helped me to become more open to working with others classmates to solve the problems that would arise in them.	I am more comfortable working with a group to solve problems than i am on my own.
7. How would you compare peer evaluation to instructor evaluation? Do you prefer one over the other?	I prefer both. The instructor can give you the overall evaluation of how your doing at that point in the year. A peer can give you the straight forward answer of how you project is compared to others.	I prefer instructor evaluation because peer evaluation has a tendency to be less honest. In this course I feel that my peers gave me some good feedback but even I must admit to being more delicate with my evaluations and ended up being more of an encourager than a giver of positive feedback. Which it is fine to be nice about things but sometimes you do need a little brutality to get the job done.	I prefer instructor evaluation because sometimes I felt my peers didnt care as much to give constructive feedback.	i actually dont prefer one over the other. The instructor evaluation is cool and the peer evaluation is cool too. The thing i like most about the instructor evalustion is you get some insight on if you met his expectations or not. Then the peer evaluation is good because you can get feedback on what they think you need to improve on.	I like both of them because the instructor would give us some feed back and the students would also give feed back or give ideas in which would make the application better.	I prefer instructor evaluation because it helps the instructor to be informed of how he/she can improve their teaching habits and routines.	I believe that peer evaluation is less reliable than instructor evaluation. I prefer instructor evaluation.
8. Is there anything that you would change about the design of the course?	No	No	No	No	No	No	Yes

Please explain.	I think the course was set up great with an instructor who showed constant interest and always gave us new ideas to consider.	I really wouldn't change anything. I love the ability to freely develop and do whatever comes to my mind. Restricting people only holds the mind back and students just don't need anymore of that.	I liked how it was laid back and move at your own pace but with some structure.	I really like the way this course is design. i would like to see more group projects instead of individual projects.	I would not really change anything about the course. I thought it was perfect the way it was.	I wouldn't change anything about the design of the course.	I think the students should have more say in the grade they receive rather than just being assigned a letter grade based on what the instructors have observed. They may
9. How is Computer Science relevant to your major?	I am a computer tech major, but I am sure that I will take another CS class along the way.	Stating that it is Medical Technology says it all right there. It will give me a better understanding of the devices I will use.	Not much directly except for how to work with computers.	I actually have an undecided major. When i first started this class i was wanting to build computers then i decided to change it because im not that good at math.	Computer Science is relevant to my Telecommunication major because we use a lot of computer programs and it is nice to understand how these programs work.	Computer science is important in the fact that it is involved in everything and therefore is relevant to my major.	I am a computer technology major so it is very relevant.
10. How is Computer Science relevant to your personal life?	My personal life doesn't really involve in programming but it may later when I have a career.	I love computers. I've always loved computers. I have literally been on a computer since I was a baby. It is always going to be relevant to my life because for the most part many components of computer science are my life. I simply enjoy it.	I would use computer science for entertainment.	I am on a computer all the time and want to learn different things about computers that i didnt know before. So i feel like i need to know as much as possible about computers.	I use computers everyday and computer science just furthers my knowledge of computers.	Computer science is involved in my personal life every time I do anything with technology. Every time I get on the internet or do anything with my computer I'm dealing with computer science.	I have always been interested in computer science since i began exploring into the computer world with the purchase of my new computer. Ever since then i have always wandered about coding and what ran the apps.
11. What skills do you think are important for computer scientists?	They must have the ability to work as a group, to be patient when something is hindering them, and be a great problem solver.	Definitely critical thinking. You seriously have to be able to analyze what you are doing and think about who or what your are doing it for. Being able to think about all the details will certainly help anyone for that matter but it is important for a computer scientist to be able to take that step back and rethink what they've done.	Logical thinking	You have to be able to understand the basics about software. I feel like if you can understand how software works than you can succeed in computer science. I feel like computer scientist do alot with computers and if you can know as much as possible with computers than computer science will come easy.	I think an important skill for computer scientists is that they have to be able to work with other people to solve problems.	I believe that computer scientist have to have a high understanding of mathematics to be efficient as a computer scientist.	Coding skills along with other basic PC knowledge
12. Did this course change the way you think about Computer Science?	Yes	Yes	Yes	Yes	Yes	Yes	No

Please explain.	Coming into college, I wasn't exactly sure what the difference between a CS major and a IT major. This made me appreciate the CS major more and I definitely hope to take more classes on this material.	It gave me a better insight into the development of programs and applications which is something I didn't truly know before.	sort of, I looked at computer science as something only nerds could do but it is actually rather simple.	Now i have a much better understanding of what computer science is about. I now know that computer science takes alot of time to understand the material.	I never really understood what Computer Science was until now.	I began thinking that Computer Science wasn't a very big deal but in the end I realized that Computer Science deals with many different aspects of the world that we live in today.	I have the same view as before
13. What did you think about App Inventor for Android?	The App inventor worked great for a new software. There were a bugs here and there but that is to be expected. They were always updating it which was good and it was very simplistic to work with.	I loved it, when it cooperated.	I enjoyed using it because of how user friendly it was.	i believe that app inventor was very complicated at times. i had some problems trying to get the right blocks in place. i did get alot better with the blocks.	I think App Inventor is a user friendly and is very easy for just about anybody to pick up and understand.	I really like the product and the features of it. I didn't like that there wasn't an undo function that accompanied the autosave feature.	I thought that it was very user friendly and that basically anyone can use it as long as u can learn the topics.
14. What was your favorite aspect of using App Inventor for Android?	The way that the software communicated with the phone was very interesting. The way the barcode scanner would use the QR code to make the app was different.	The blocks editor. It was a little frustrating but it did make the thought process way easier.	how easy it was.	My favorite aspect using app inventor was being creative with the projects. i was able to come up with ideas that no body else could come up with.	My favorite aspect of using App Inventor was that the applications you create actually work for the phones and that is really cool to know that you created this app.	My favorite part of App Inventor was packaging it for the phone and scanning the QR codes.	I liked how customizable we could make our apps
15. What was your least favorite aspect of using App Inventor for Android?	The way that the bugs would some time trip you up and mess with you when there was no apparent problem.	The design view. I'm a very visual person and I don't like the lack of control I had when creating the layout of an app. I suppose it was not the most important thing to worry about but I like to make nice looking things.	the amount of time it took to get something to work.	My least favorite was building the blocks. i felt like it was extremely challenging to come up with the exact blocks that your project needed to work.	My least favorite part about App inventor was that sometimes how your preview of the app looks does not always look the same when you build the program on the phone.	My least favorite aspect of App Inventor was the autosave feature and that there was no sort of undo feature.	All the changes that occurred during the semester messed with some of our projects

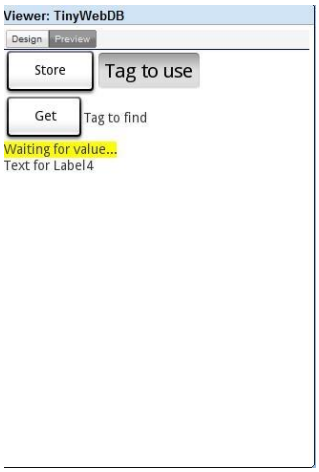
APPENDIX M:

Student's Projects

Below is a sampling of student projects submitted during the course at various stages.

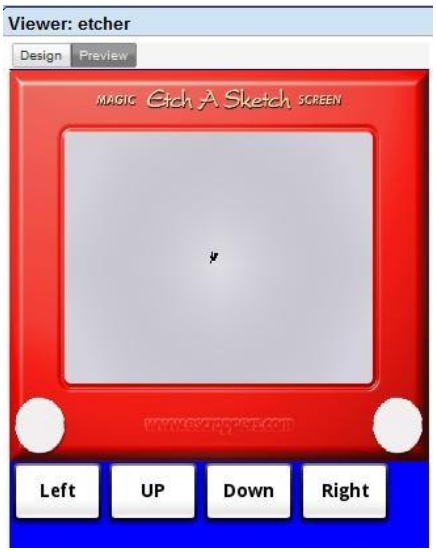
Text to Database Project:

This is an application that allows the user (while in the application) to receive a text message and be able to store on an internet database. The user will be able to store their message under whatever tag that they want to, as well as be able t find and pull what tag the want to. This application is fairly simplistic and user friendly, which was the whole motivation behind it.



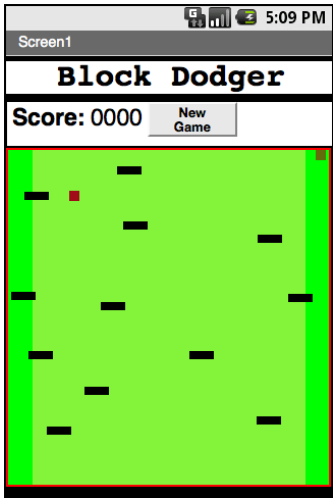
Etcher (Etch A Sketch) Application:

This application is based off of the old Etch A Sketch drawing toy. There will be a black ball in the middle of the screen that the user will be able to control with the directional buttons. The ball will draw solid black lines. Enjoy!



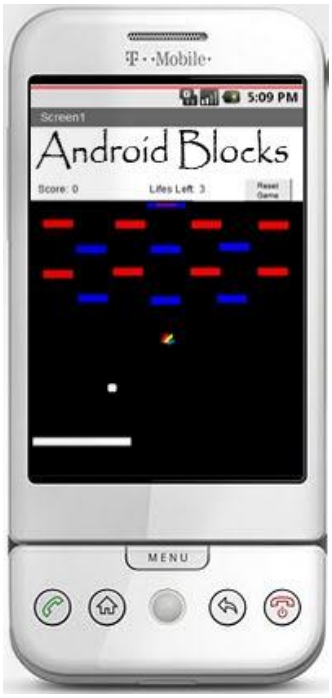
Block Dodger:

This entertaining application can be played by any one. Tilt the phone back and forth to catch the red dot, but be careful to not touch a black block or the edge. Try it out for yourself and see how high of a score you can get, don't forget to dodge the wall and blocks!



Android Blocks:

This app was created out of the old block buster game. This type of game was one of the first major games on cell phones. The game is to keep the ball bouncing off of the blocks. When the blocks get hit they disappear, the ball speeds up, and it adds points to your score. You use the touch screen to control the bottom bar to keep it from going off the bottom of the screen. When all blocks are off the screen the screen will reset with all the blocks on it again.

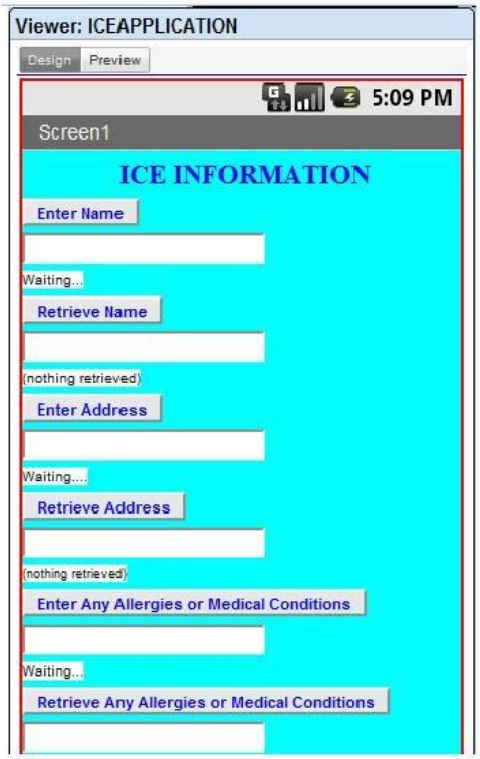


Marauder:

This is a barcode scavenger hunt game. It uses the barcode scanner to collect QR codes with number values. The only catch is if you make your own codes you must make the numbers decimal numbers or your normal numbers must not repeat. This is a built in method to avoid cheating as every code scanned is always compared to the list it is creating as it goes. This game also has the functionality to use TinyWebDB by uploading high scores to a leaderboard. This may be used when a game is created online for players. So one could hide QR codes all over a website and then send people on a virtual scavenger hunt. Knowing the highest total possible to achieve on a set game would allow for the leaderboard to be used properly.

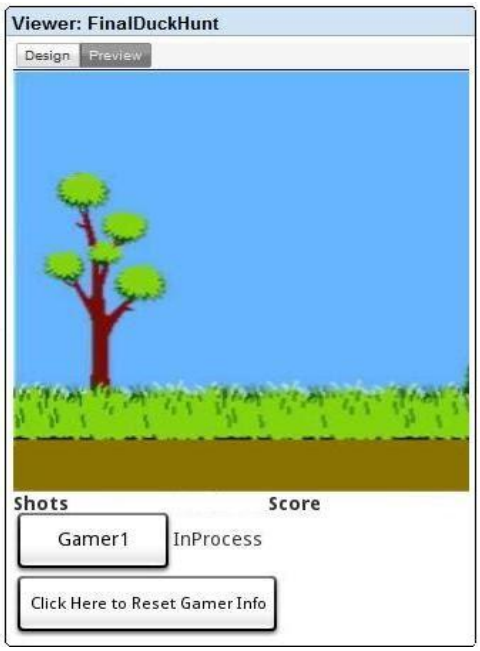
ICE APP:

This Application is a lifesaver. All you have do to is input your emergency information including your allergies and address. After input if any emergency arises anyone will have access to this important information.



Duck Hunter:

This application is an Android recreation of the original NES game, *Duck Hunt*. It is a simple game of duck hunting. A duck will fly across the screen and the user touches it to fire a shot and make it disappear. Every duck shot is 10 points and the score can be uploaded by the user to an online database where they can compete with other players. Happy hunting!

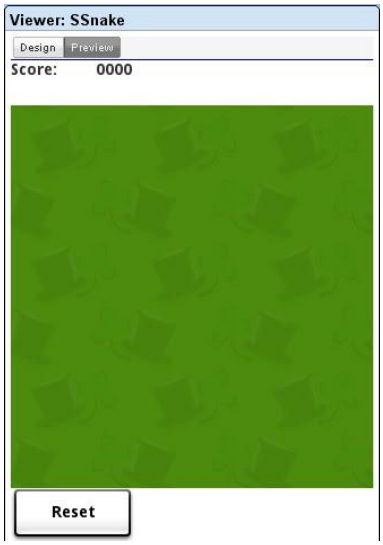


Bird Watcher:

This is an app that appeals to those who love enjoying the outdoors in Indiana. There are eight pictures of different birds you may see in Indiana. You can click on the picture to hear the sound, see a bit of displayed information about this bird, and you will have its name shown. Check boxes are also located beneath each picture so you can mark it after you have spotted it.

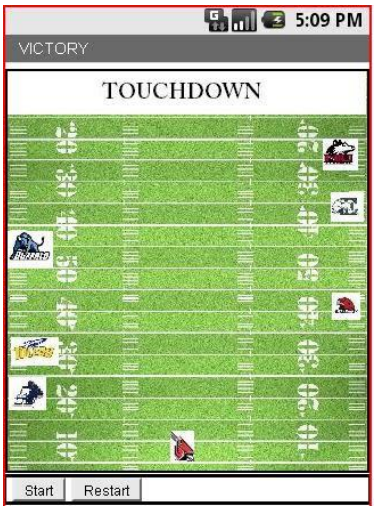
Snake Game:

This application is based on the standard snake game but instead of growing longer it grows faster. The goal is to move the snake using the orientation sensor to eat the snake food which is the blue dot be careful though and run into the screen edges.



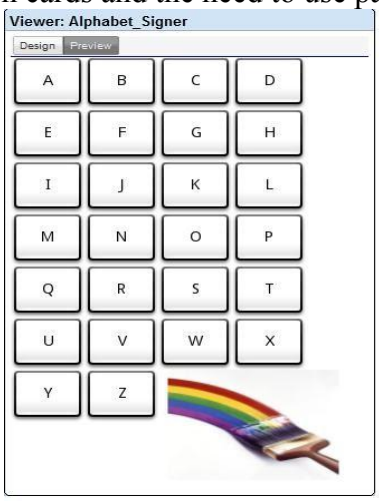
BSU FOOTBALL:

This application gives the user the chance to help Charlie the Cardinal Navigate his way down the field past the opposing MAC Teams. You simply Press Start and drag the BSU logo down field, past the opposing MAC teams and make your way to the endzone. But be careful, coming into contact with another team has it's penalties. So take the challenge and lead Charlie the Cardinal to Victory!



AlphaSign:

This is an application that teaches the user how to say the alphabet using sign language. The application has a user friendly design that is very simple to use. The screen has the 26 letters of the English alphabet and when a letter is clicked the signing of the letter is displayed in the bottom lower right corner of the screen. It is an app that anyone can use and is great for road trips and is perfect for the entire family. The app also replaces the hassle of flipping through flash cards and the need to use paper to make them.



---

31 Flavors:

I did this app because earlier this year, I went to B And R and had no idea what the flavors were and what was in each one and whether or not I would like them. So I got rushed and went with vanilla. I was hoping that this app would solve that problem so that future ice-cream cravers will not have the same dilemma I did. This app would randomly select the flavors for the user.

PumpkinHead:

It's a simple app that is designed as a holiday Mr potato head, or a face that you can change each feature. The idea of the application is to be as simple as possible so that anyone from adults to children could play with it. From a design view, I didn't want a bunch of buttons to represent each piece, so instead you can touch the eyes, nose, or mouth to cycle through to the next available piece. You can also drag a piece to move it around. One button called the color button would change the colors. Instructions are placed at the bottom to help the ease of use. The app also has what I would consider Easter egg type things such as if you drag a piece you will hear a wicked laugh, and if you shake the phone it will make all the pieces random.