BLANK

GROUP COLLABORATIVE COMPUTER PROGRAMMING

WITH THE AID OF A ROBOT:

DISCOVERY-BASED LEARNING

A Dissertation Presented to the
Faculty of the College of Education
University of Houston

In Partial Fulfillment
of the Requirements for the Degree

Doctor of Education

By

Maria T. Earle

May, 2011

SIGNATURE PAGE

Acknowledgement

All work is collaborative. This process began some four years ago with my requirement to be successful in a doctoral program. I sought advice from Dr. Diana Dale who provided answers and encouraged me to pursue the degree. Thank you Dr. Dale for expressing your confidence in me to succeed.

And thus the journey began which sometimes required a two-hour, often rushed commute to class. Upon arriving, Ms. Gussy, the parking lot attendant, often greeted me with her cheerful, welcoming smile. Thank you Ms. Gussy for your words of encouragement throughout the years.

To my esteemed dissertation committee, Dr. Melissa Pierson, Dr. Mimi Lee, Dr. Allen Warner and Dr. Farrokh Attarzadeh , I sincerely appreciate your time and support. I would especially like to thank my chair and advisor, Dr. Pierson, for encouraging me through all my various research ideas with such patience and sophistication.

I owe a debt of gratitude to my reviewers, Mrs. Bernice Earle, Mr. George W. Earle, Dr. Gladys Johnson, and Mrs. Mabelle Thompson. I sincerely appreciate your time and efforts and support.

Next, I would like to thank my family. To my husband, George W. Earle, thank you for your unfailing love and challenging me to be the best I can be. Thank you Mrs. Bernice Earle for supporting me throughout this process. My reason for being are my daughters. Thank you Asha, Shaina, and Nadia for being my spark, for being my

teachers. I would also like to thank my sisters, Madeline, Jacqueline, and Justine for standing by me and giving me encouragement. None of this would have been possible without my parents, Mr. Robert and Judge Mamie Chinn. Thank you for the love and support you have shown me throughout my life.

Finally, I owe a debt of gratitude to all my study participants, particularly my interviewees, Tanya, Kathy, Todd, Anne, and Elaine. This study was undertaken with the idea some three years ago of how might a neat little robot help my students gain an understanding of basic computer programming. I hope you found the experience valuable. Moreover, I hope you appreciate your ability to work in an ill-structured discovery learning environment and succeed.

Above all, I give all praise and honor to the enduring Holy Spirit.

GROUP COLLABORATIVE COMPUTER PROGRAMMING

WITH THE AID OF A ROBOT:

DISCOVERY-BASED LEARNING

A Dissertation Presented to the
Faculty of the College of Education
University of Houston

In Partial Fulfillment
of the Requirements for the Degree

Doctor of Education

By

Maria T. Earle

May, 2011

Earle, Maria T. "Group Collaborative Computer Programming with the Aid of a Robot: Discovery-based Learning." Unpublished Doctor of Education Dissertation, University of Houston, May, 2011.

Abstract

The notion of discovery-based learning began in the early sixties (Bruner, 1961). Discovery-based learning has been considered an effective form of inquiry instruction particularly in a group learning environment because it allows students to construct knowledge by learning collaboratively from each other, become more active in the learning process, and gain ownership of their own learning (Bruner, 1961; Barkley, Cross, & Major, 2005; Bell, 2003). This type of learning is not only meaningful learning it promotes lasting knowledge (Bruner, 1961; Barkley, Cross, & Major, 2005; Denzin & Lincoln, 1998). However, its effectiveness remains controversial (Louis, 2010; Ramadhan, 2000).

While teacher-led pedagogies dominate computer-programming research literature, this study sought to explore the influence of a group discovery-based pedagogy. This study provided a bridge between learning about computer programming and discovery based learning. Additionally, this study sought to expose the benefits of learning about computer programming with the aid of a novel learning platform, a robot. While working in small groups, forty students of two sections of an introductory computer course were observed and videotaped during a two-week computer programming learning activity. The goal of the learning activity was to have student

groups “discover” computer programming concepts while attempting to program a robot to solve a problem.

The primary sources of research data were 900 minutes of group observation videotapes, over 100 participant interview transcriptions, and participant reflection papers. Carspecken’s (1996) methods were used for data coding. Data analysis netted five emergent themes which defined students’ characteristics and behaviors from learning in such an environment, namely, (1) frustrations from engaging with the learning aids, the robot and its remote controller, (2) frustrations from trying to understand the subject matter, computer programming, particularly modular programming, (3) prematurely celebrating success because they did not know what success really looked like, (4) not staying on course of the stated lab objective because they wanted to be more creative, to be challenged, and (5) dealing with adversity in team mates’ work ethic while constructing computer programming knowledge collaboratively.

Based on findings and literature, this study provided strategic recommendations on how to mitigate the five aforementioned behaviors to further enhance learning in a group-collaborative discovery learning environment. Findings and recommendations from this study could be used by educators to implement a discovery-based pedagogical environment in their classroom, perhaps as a first course in computer programming. Findings could also provide a framework for future research. What is unknown is how much knowledge actually transferred in this discovery-based knowledge transference environment? Further research might provide validation of the effectiveness of the use of

a group-collaborative, discovery-based knowledge transference environment on learning outcomes.

TABLE OF CONTENTS

APPENDIX

LIST OF TABLES

LIST OF FIGURES

CHAPTER ONE:

INTRODUCTION

Graham, a leading historian on American education (Harvard, 2010), chronicled a 100-year span of the American school system by segmenting the span into four phases, namely, assimilation, adjustment, access, and achievement (Graham, 2005). In each phase, a discussion of societal educational expectations in association with national needs during the twentieth century ensued.

Background

The Assimilation Phase occurred during the first two decades of the 20th century. During this phase, new immigrants had arrived at the American border. The American culture was mostly agrarian and skills were needed, for the most part, to tend to farms where such skills were learned. Schooling expectations were at a minimum. The focus of the society was to have these new Americans serve the ideals of the nation, which at that time were to have homogenous holistic patriotic citizens who would most likely work on farms.

The next phase occurred from 1920 – 1954 and was known as the Adjustment Phase. During this phase and particularly after the depression, expectations of schooling increased, prompted mostly by wealthier Americans. The expectations of schooling moved away from assimilation of groups towards satisfying the needs of the individual. Wealthier Americans expected schooling to support their desire for increased intellect

and subsequently schooling was adjusted to meet this desire. For instance, child-centered schools began to emerge during this phase, although not widespread for they were located in wealthier neighborhoods.

The Access Phase spanned from around the early 50's to the early 80's, and schooling expectations increased once again. Schooling was expected to appeal to a broader citizenry by providing schooling not only for the individual but equal access for all. Additionally during this time, pressing global initiatives such as the race to the moon and the Cuban missile crisis were increasing in scope. On the other hand, national standardized test scores were decreasing and had been for the past two decades. This dilemma caused Americans to feel "discouraged and somewhat ambivalent" towards the education system and subsequently Americans began to question the value of schooling and its ability to produce graduates with skills that may help alleviate such global challenges.

Consequently, in 1983, partly due to American's questioning of the school system, a dramatic shift occurred and thus ushered in the Achievement Phase. For the first time in over one hundred years, schooling was reformed. During this phase, schooling was expected to provide not only access to schooling for all but excellence in academic achievement for all. To motivate and assess such achievement in schools national standards, as described in contemporaneous documents such as *A Nation At Risk* and *No Child Left Behind*, were established. The thought was to calm Americans' questioning of the school system by mandating that schools support increasing students'

national test scores and graduating more students with skills that could help solve emerging global challenges.

Graham's concluding remarks on schooling expectations in America during the 20th century are that "The nation flourishes when all its citizens can participate fully in its public life and in its economic growth" and that skills engendered from schooling today "…are enormously more important …than they were a century ago" (Graham, 2005, p. 254).

The Problem

But, what are the enormously important skills for today? This study sought to address this question.

First, as a comparison, during the agrarian age, skills were needed for the most part to tend to individualized farms and thus schooling expectations for most citizens were at a minimum. However today, most Americans no longer tend to farms. They tend to computers. In today's culture, computers are pervasive for they can be found at home, at work, in school, and with the invention of the internet, on the go. Thus adequate computer skills seem important today. But what are adequate skills?

While a contemporaneous understanding of how to use a computer to say write a term paper or send an email safely on the internet are important, to be successful today computer fluency is required (Araujo, Filho, & Losada, 2005; Evans, Martin, & Poatsy, 2009; Gardner, 1999; Keller, 2008; Kolb, 2008). Computer fluency requires more than a cursory understanding of how to use a computer (Araujo, Filho, & Losada, 2005).

Computer fluency requires a deeper level of understanding how computers work (Araujo, Filho, & Losada, 2005).

Secondly, work is no longer local and fenced-in, it is global and borderless (DOC, 2011; Diamond, 2005; Friedman, 2005). Americans no longer have to depend on their individual efforts on their individual farms to secure their lifestyles. They have the opportunity to work with groups of people across the globe and partake of the available $10 trillion dollar revenue (DOC, 2011) using collaborative computing platforms (Liveops, 2011; ODesk, 2011) that allow for such endeavors.

However, working in this increasingly digitally interconnected global marketplace, collaborative skills might enhance success (Diamond, 2005; Friedman, 2005). Moreover, employers particularly value employees who can work collaboratively to solve problems with diverse people (Gordon, 2011; Kubler & Forbes, 2005). In fact, some employers mandate such skills as a condition of employment (Barkley, Cross, & Major, 2005).

Thus today, it seems prudent for schooling to support students' attainment of two skills in particular, computer fluency and collaborative skills so that students might be poised to compete more effectively in the emerging global flattened society. But how might such learning be scaffolded by schools?

The Need

A course was needed to engender such skills. However, in any course when the goal is to impact student learning outcomes, factors incumbent on learning, such as the

learning material, the learner, instructional aids, and instructional strategy, should be considered (Dick, Carey, & Carey, 2005).

Learning material.

First, the learning material for the present study was designed to facilitate students' deeper understanding of how computers function and subsequently impact their computer fluency (Araujo, Filho, & Losada, 2005). The basic function of a computer is to compute and process data. Computers compute and process data by carrying out steps detailed in a computer algorithm. A computer programmer then converts the computer algorithm into a computer program by writing code using a computer programming language (Evans, Martin, & Poatsy, 2009).

However, to write a computer program, the programmer must understand not only a computer programming language but also how the computer functions. Thus, the learning material was so chosen to focus on students' attaining increased computer fluency by having them understand the basics of computer programming.

Instructional aids and the learner.

Understanding computer programming concepts tend to be difficult for computer science majors and particularly challenging for novices (Black, 2009) such as the participants in this study. One of the more challenging parts of learning about computer programming is programming language syntax. Thus, given this reality, a highly scaffolded cognitive computer programming instructional aid requiring a low understanding of computer programming language syntax was used for the study. The

literature will provide a discussion on why the final instructional aid, a programmable robot, was chosen.

Instructional strategy.

First, in an attempt to address increased group collaborative skills, a group based instructional strategy whereby group members worked together to solve a problem was used. Secondly, to maximize student learning outcomes for these critically sought after skills (Araujo, Filho, & Losada, 2005; Diamond, 2005; Friedman, 2005; Gordon, 2011; Kubler & Forbes, 2005), the literature will also discuss why a discovery-based learning strategy was used. Briefly, in this type of learning environment, students engage with artifacts, learning materials and group mates to solve a problem without the benefit of prior knowledge and/or teacher instruction and in the process discover learning concepts (Bruner, 1961). The literature will also discuss why learning in this way provides for meaningful lasting learning.

Before implementing a new course, a model should be consulted to inform on possible course enactment and outcomes (Bybee, et al., 2006; Linn, Sloane, & Clancy, 2006; Swan, Hofer, & Swan, 2011). Thus, taking the aforementioned learning factors into consideration, a search for literature that might shed light on how such an environment might unfold ensued. Unfortunately, there was sparse research discussing classroom enactment and findings on the effectiveness of group collaborative computer programming within a discovery-based learning environment. Most computer programming learning occurs in individualized, teacher-led instructional environments (D'Souza, Kazlauskas, & Thomas, 2009).

Thus, the ultimate need for this study became to investigate *how* (emphasis added) students engage in group-collaborative computer programming in a discovery-based learning environment. Specifically, the following research questions anchored this study:

Research Question One: What interactions and computing behaviors exist when student groups engage in collaborative computer programming in a discovery based learning environment with the aid of a robot?

Research Question Two: What are students' perceptions on their robotic group collaborative computer programming experience?

Definition of Terms

Algorithmic Thinking: Algorithmic thinking is a way of systematically thinking about what procedures a computer must execute to complete a task (Dale & Lewis,

2007). Participants in this study had to think algorithmically in determining how to complete a flowchart which would depict how the robot would complete a task, such as bowling.

Collaboration: Cooperative activity characterized by dialogue and shared effort (Warner, 2008). Participants in this study cooperated to collaborate on how to get a robot to solve a problem.

Collaborative Learning: Learners work together to gain shared understanding of a concept and in the process each members' understanding of all the material is deepened (Weinberger, Ertl, & Fischer, 2005; Barkley, Cross, & Major, 2005). Participants in this study collaborated on every learning task of this study. In the process, the goal was to gain a deeper understanding of the learning material because all learning tasks were shared learning tasks.

Computer fluency: A deeper level of understanding how computers work (Araujo, Filho, & Losada, 2005). Instead of a mere cursory understanding of how to use a computer to say produce a document, the emerging citizenry needs to understand how computers work (Araujo, Filho, & Losada, 2005; Keller, 2008).

Computer literacy: Knowledge and skills needed to use computers and technology efficiently (Gardner, 1999; Keller, 2008).

Conceptual Understanding: Conceptual understanding is a type of intellectual skill in which a learner must apply apriori knowledge to understand some event (Smith & Ragan, 1999). Participant's conceptual understanding of computer programming was

limited in this study. They were novices and did not have much apriori knowledge. Furthermore, in understanding the event, computer programming, one must understand abstract concepts (Black, 2009). Thus, cognitive loads were minimized by working with an instructional aid that scaffolds understanding of abstract concretes with a concrete aid, a robot.

Constructivism: Provides a way for educators to allow students to actively construct knowledge by interacting with artifacts in the learning environment (Duffy & Cunningham, 1996; Howland & Jonassen, 2003; Schunk, 2004). Constructivism was in play in this study. Participants in this study interacted with a robot and its accessories to construct knowledge about computer programming.

Controller Programming: Use of a handheld remote controller to program a computerized robot. Traditionally, computer programming is completed via a keyboard. However, in an attempt to engender a low cognitive processing learning, a non-syntaxed computer programming approach to programming was sought. What resulted were a robot and its non-syntaxed, script-like programming environment. A programming script was first designed on paper, and then entered into the robot's computer via a handheld remote controller.

Controller-scripted code: Code consisting of remote controller positional movements written in a script-like language. This type of coding environment was used to forego an understanding of traditional computer programming language syntax which traditionally is entered with a keyboard.

Cooperative Learning: Work is divided among group members and each member understands one concept and then all report to the group (Barkley, Cross, & Major, 2005). Although participants had the option of assigning work assignments, none did so. All worked collaboratively.

Discovery-based learning environment: Environments where students' actively problem solve by "manipulating, exploring and investigating", not passively reading, or listening to a teacher (Bruner, 1961; Schunk, 2004). This was not an instructor-led knowledge transference environment. Group mates tinkered with the robot and its accessories, viewed online functional videos, and engaged in discourse with group members to learn about robotic computer programming.

Gamer: Players of video games (Wikipedia, 2010). The remote controller used in this study is similar to the remote controller used in popular video games today. The average age of a gamer today is around 34 years of age (ESA, 2011).

Group Collaborative Learning: Collaborative learning, but within a group consisting of more than two members who work together to gain shared understanding of a concept. All participants in this study participated in group collaborative learning.

Information Processing (IP): This definition was added for comparison purposes. IP deals with the neurophysiology of learning. It provides an explanation for the structure of "workflow" mental processes that occur during learning (Schunk, 2004). This study did not attempt to determine how students might learn about computer programming by analyzing internal learning mechanisms within a discovery-based

learning environment; it sought to determine how students might learn by analyzing external scaffolds, namely group mate interaction and the robot, within a discovery-based learning environment.

Learning Outcomes: Learning outcomes are skills the learner displays once they have obtained understanding about conceptual concepts (Mayer, 2001). In this study, the objective was to learn how to program a robot to solve a task. The learning outcome was verified with a successful executable run of modular programming code.

Paired Programming: Two members work together to produce a final computing product (NCSU, 2010). In a paired-programming environment, two members typically work cooperatively to solve a computer-programming problem. A problem is divided into pieces and assigned to each member. Then each writes code to solve their individual problem and then reports to their group.

Student Centered Learning: An environment where the problem to be solved is not well prescribed, it is purposely "ill-defined" and additionally students have had no prior knowledge of the subject matter (Jonassen & Land, 2000). In this study, even participants had no prior teacher-led instruction on how to go about solving the problem.

Tactile Computer Programming Learning: Programming a computer by using a remote handheld controller, not a keyboard. All programming in this study was facilitated with a remote controller.

Teacher Centered Learning Environment: An environment where the problem to be solved is based on prior instruction, given by a teacher (Jonassen & Land, 2000).

This study did not center take place in a teacher-instruction environment. This study utilized a student-centered, discovery-based learning environment.

Transference: Knowledge transferred from someone or something outside the learner to the learner (Jonassen & Land, 2000). Knowledge was not transferred from someone or someone outside the learner, knowledge was discovered within the learning group.

Justification for Educators

In an introductory computing course, during the fall of 2008, the researcher introduced a group robotic computer programming project for the first time. With minimal teacher-led instruction and observation, student groups successfully programmed the robot and anecdotal assessments pointed to learning. However, the researcher wanted to determine how students might engage with group mates by more closely observing their group behavior in a discovery-based learning environment.

Thus, the current study sought to determine how students might learn about computer programming in a group-collaborative, discovery-based learning environment. In addition, the researcher sought to identify and describe some defining characteristics that these novice students brought to their programming experience. Identification of such characteristics may inform educator's further consideration for group collaborative discovery-based learning environment.

Next, this research used a novel approach to learning about computer programming, a humanoid robot. Novel technology can suffice as a motivational tool in

learning (Kolb, 2008) which might in turn reverse the decline in computer and technology courses (Rodrigo & Baker, 2009).

Justification for Research

Most research on computer science is quantitative in nature. Secondly, this research tends to be set in teacher-led, individualized learning environments. While quantitative research methods provide valid information on a variety of research questions, the qualitative methods used in this study allowed for a rich holistic description of how student groups engaged in a discovery-based learning environment to learn about computer programming.

Next, this research used a novel learning aid to help students understand computer programming, a robot. Use of robots as educational platforms are showing promise (Arango, Altuger, Aziz, Chassapis, & Esche, 2008). When new technology becomes available that shows promise on influencing learning outcomes, it is important to scrutinize methods that might best expose advantages from using such technology (Leonard, 2002).

Limitations

First, the goal of this research was not to determine computer programming learning outcomes or to generalize findings to all computer courses. The goal of this small case study was to describe some defining characteristics of how students might interact in a group collaborative discovery-based learning environment to learn about computer programming concepts during one semester at a community college. Such

characteristics could then subsequently be used by educators to inform on implementation of such computer programming strategy in their classrooms.

Secondly, the programming environment was limited. One of the more difficult tasks in computer programming is understanding computer programming language syntax. To scaffold this understanding, a non-syntaxed programming environment was sought. What resulted were a robot and its non-syntax, script-like programming environment. While programming the robot allowed participants to work together to gain an understanding of some of the more basic programming concepts such as algorithms, flowcharting and modular programming, exploration of more extensive computing concepts such as sorting and object oriented programming were not available. However, learning of such advanced concepts was not the goal of the study.

Conclusion

In conclusion, this chapter has 1) set this study in context by providing background information for the study, 2) discussed the research problem and questions, 3) discussed the need for research, and finally 4) provided research justifications and limitations. Next, chapter two will discuss relevant literature bearing weight on this research, particularly learning theories, issues in computer programming, and finally today's college student in relation to computer fluency and collaborative skills.

CHAPTER TWO:

LITERATURE REVIEW

The phenomenon this study sought to investigate was *how* students engage in group-collaborative computer programming in a discovery-based learning environment. Specifically, the following research questions anchored this study:

Research Question One: What interactions and computing behaviors exist when groups engage in collaborative computer programming in a discovery based learning environment with the aid of a robot?

Research Question Two: After completing the collaborative computer programming activity, what are students' perceptions on their robotic group collaborative computer programming experience?

In this chapter, you will not find a bevy of literature discussing group-collaborative computer programming learning in a discovery-based environment. Hardly any exists. However, in order to understand how students might work together in such an environment, this chapter will discuss those factors bearing weight on this study by 1) providing background information on how one learns in general, 2) discussing computer programming learning 3) discussing findings from literature in which collaborative-like computer programming was studied, and finally 3) a discussion of today's learner in a community college setting in relation to societal needs.

Learning Theories

There are over five hundred learning theories (Leonard, 2002). A learning theory provides a framework for learning and the transmission of knowledge. It does so by providing not only an explanation for learning (Unger, 2007), but also a set of principles that can be modified as research uncovers findings (Schunk, 2004).

A learning theory can address learning in a variety of ways including how learning occurs, what factors influence learning, the role of memory and motivation, which processes are involved in self-regulation, and what the implications are for instruction (Schunk, 2004). Another way of categorizing learning is by how knowledge is transmitted (Schunk, 2004; Ginsburg & Opper, 1969; Alessi & Trollip, 2001; Gates, 1999; Plomp & Ely, 1996; Jonassen & Land, 2000; Smith & Ragan, 1999) and the most dominant American education philosophy to date, essentialism, states that the primary purpose of education is the transmission of selected information from one generation of learners to the next (Warner, 2008).

This study looked at the transmission of computer programming knowledge.

Knowledge transference.

Knowledge transfer can be facilitated by either external and/or internal factors. External transference can occur when some external factor, such as someone or something outside the learner, influences learning (Jonassen & Land, 2000; Unger, 2007). An example of an external knowledge transference theory is behaviorism. It claims that learning is impacted when knowledge is transferred to the learner by repeated

external stimuli presented to the learner during the learning process (Plomp & Ely, 1996; Alessi & Trollip, 2001; Gates, 1999).

On the other hand, internal transference occurs when the learner relies on internal factors to impact learning such as their own inner processing. An example of an internal learning transference theory is cognitivism. Cognitivism claims that learning is impacted when one uses their own inner mental strategies to gain knowledge (Alessi & Trollip, 2001; Smith & Ragan, 1999).

Since this study was concerned with how external factors, namely pedagogical environments and group members might influence one's learning, the rest of this section will discuss knowledge transference in association with these components.

Knowledge transference and pedagogical factors.

Discovery-based learning is not the same as inquiry-based learning. First, inquiry-based learning necessitates apriori knowledge (Swan, Hofer, & Swan, 2011). Secondly, this knowledge was most likely transferred in a teacher-lead environment (Swan, Hofer, & Swan, 2011).

Bruner's (1961) book, The Act of Discovery, ushered in the notion of discovery-based learning. Since then, the effectiveness of discovery-based learning has been questioned. Does discovery-based learning add to learning or detract from it (Louis, 2010)? What aids should be utilized to help novices learn in a discovery-based learning environment (Ramadhan, 2000) without threatening the discovery aspect of learning? Nonetheless, discovery based learning has found a large following in the education

community lately (Jonassen & Land, 2000; Pierson, 1999). This study sought to add insight to discovery based learning theories.

Learning should be meaningful. Meaningful learning necessitates knowledge transference via construction of knowledge (Swan, Hofer, & Swan, 2011). In discovery based knowledge transference, environment students engage with artifacts, learning materials and group mates to solve a problem without the benefit of prior knowledge and/or teacher instruction (Bruner, 1961). In the process of constructing knowledge, students discover learning concepts (Bruner, 1961).

However, when constructing knowledge in a discovery environment, the learner will most likely struggle and become frustrated (Duffy & Cunningham, 1996). Nonetheless, struggling in this way not only supports meaningful lasting learning (Barkley, Cross & Major, 2005; Denzin & Lincoln, 1998; Duffy & Cunningham, 1996), it provides an avenue for innovative outcomes particularly with teams (Fagerberg, Nelson, & Mowery, 2005). Combining a discovery-based learning environment with group computing can not only allow for social transformative relationships by allowing groups to problem solve (Thurlow, Lengel, & Tomic, 2004), but can also result in learning transformations (Bonk, Lee, Kim, & Lin, 2009; Michaelsen, Knight, & Fink, 2002).

Knowledge transference and group learning epistemologies.

Group work is focused on achieving a group goal (Bothamley, 2002). Groups can work cooperatively or collaboratively. Cooperative epistemologies came out of the

seventies in an attempt to move away from the traditional individualized learning (Barkley, Cross, & Major, 2005; Michaelsen, Knight, & Fink, 2002; Pederson & Digby, 1995). In a cooperative environment, work is subdivided among group members and each member reports to the group on their individualized findings/learning (Barkley, Cross & Major, 2005). Knowledge is transferred in a piecemeal fashion. An example of a cooperative based epistemology is project-based learning. In a project-based learning environment, students study well established structured cases (Shank, Berman, & Macperson, 1999).

On the other hand, in a group-collaborative environment, the group works toward a common goal and in the process each members' understanding of the learning material is shared, it is not taught (Weinberger, Ertl, & Fischer, 2005; Barkley, Cross, & Major, 2005; Gokhale, 1995). In essence, learning is discovered. An example of a collaborative epistemology is problem-based learning (Nelson, 1999). In problem-based learning students attempt to learn by solving "ill-defined" problems which not only lack structure but also lack traditional teacher presence (Shank, Berman, & Macperson, 1999; Jonassen & Land, 2000; Squire, 2007). However, problem-based learning helps engender collaborative skills (Gale, Wheeler, & Kelly, 2007).

When problem-based learning occurs in a group, social capital scaffolds knowledge transference (Jonassen & Land, 2000). In essence, learning is expected to be discovered (Bruning, Schraw, Norby, & Ronning, 2004; Jonassen & Land, 2000). Learning in this way promotes meaningful, lasting learning (Barkley, Cross, & Major, 2005; Denzin & Lincoln, 1998), by allowing students to problem solve by actively

learning, not passively reading or listening to a teacher (Schunk, 2004; Squire, DeVane, & Durga, 2008). However, when groups come together to work on a task, they should have a belief that they can be successful, or else they will not be motivated to engage with group members to solve the problem (Bandura, 2000).

Knowledge transference and group member attributes.

Group knowledge transference can be influenced by either internal group mate attributes or external attributes. An example of an internal attribute is discussed with the fundamental interpersonal relations orientation (FIRO) theory. This theory looks at groups in the guise of interpersonal group behavior (Rudestam & Newton, 2007). The claim of the FIRO theory is that group member's interaction in a group is predicated by three internal needs, namely, inclusion, control, and affection (Tannenbaum, 1959). However, this study was not concerned with how one's individual needs might influence knowledge transfer. It was concerned with how external factors, group mates and a robot might influence computer programming learning.

Hackman, known for his work in social and organizational psychology (Harvard, 1999) discussed group work in a different vane; he described components of a successful group environment given attributes of group members (Hackman, 1990). He stated that small group-based learning is thought to be effective in just about any course and can be transformative with sufficient effort, that members can span the age gap from 18 to 80, that groups do not need an academic superstar to do super work, and that groups help individual members of the group better understand the material.

Additionally, Hackman stated that in order to be effective, "members in groups need to exhibit the following attributes:

1. Adequate knowledge and skill to bear on the task.
2. Task performance strategies that are appropriate to the work and to the setting in which the work is to be done.
3. Motivation to exert sufficient effort to accomplish the task at an acceptable level of performance (Hackman, 1990; Michaelsen, Knight, & Fink, 2002)".

In summary, as opposed to the one-way monologue of a teacher-led environment, group knowledge transference can provide an opportunity for group members to engage in a dialectical process that supports minimizing ignorance which eventually leads to a more informed understanding (Denzin & Lincoln, 1998) in a discovery based learning environment.

To summarize, this section has discussed how external factors, working in a group collaborative discovery based learning environment can promote maximal meaningful learning; however, since computer programming is a cognitive learning task, internal factors as described by Weinstein and Mayer may help with understanding computer programming. The next section will discuss the study's topic area, computer programming.

Computer Programming: Defined

Computer scientists use software to solve real world problems (Hoganson, 2008). The power of computers lies in their speed and processing power (NSF, 2008). This power can be maximized with efficient computer algorithms. The first step in computer programming is to devise an algorithm. A computer algorithm states the logical processing a computer must undertake to solve a problem and consists of step by step descriptions of tasks the computer must perform to complete a goal (Dale & Lewis, 2007; Evans, Martin, & Poatsy, 2009; Microsoft, 2002).

While humans use a range of algorithms everyday from the mundane, like tying shoe laces to the not-so-mundane, such as deciding on an academic career. However, algorithmic thinking combined with computers allow utilization of the computer's power and processing speed to help problem solve "real-world" problems (Dale & Lewis, 2007; Hoganson, 2008)

The next step is to have a computer programmer, using a programming language, code. However, one of the more challenging aspects of computer programming is computer programming language's abstract concepts, such as its programming language syntax (Black, 2009; Forte & Guzdial, 2005). The abstractness of these concepts may not only prove challenging for experienced students but particularly for novices (Black, 2009; Postner, 2003; Forte & Guzdial, 2005).

Computer Programming as a Cognitive Activity

However, while these external factors may prove effective, learning how to computer program is largely an internal cognitive task (Smith & Ragan, 1999). Gagne

(1985) felt that human behavior was complex, and given such devised several categories of human learning; one is cognitive learning (Smith & Ragan, 1999). Weinstein and Mayer (1986) went on to state that individuals have the capability to impact cognitive learning by engaging in cognitive strategies by either rehearsing, elaborating, organizing, metacognition and modifying, and affective motivational strategies. As discussed previously, cognitivists claim that learning is scaffolded by one's inner mental strategies (Alessi & Trollip, 2001; Smith & Ragan, 1999) and thus learning a cognitive subject relies on internal learning transference.

Group Programming – Paired Programming

Although computer programming learning is thought to be best facilitated by internal cognitive working, use of paired-computer programming learning transference was endemic in literature on group computer programming. With paired programming, two members work together to produce a final computing product (NCSU, 2010). However, the work is used completed cooperatively, not collaboratively.

With paired-programming, more often than not, each member has been assigned a role or task of some sort after learning in a teacher knowledge transference environment (Williams & Upchurch, n.d). Essentially, paired-programming is a type of cooperative programming knowledge transference environment. As discussed in the previous section, a cooperative learning environment is not the same as a collaborative environment and the former promotes meaningful shared learning (Barkley, Cross, & Major, 2005).

The search for the group computer programming studies focused on reviewing articles in the digital American Computing Machinery (ACM) database, and the IEEE XPlore databases. Fifty articles written over a ten-year span were initially reviewed.

Almost all of these studies found that paired programming not only produced higher quality computer programs, but also students enjoyed the experience and had their self-confidence boosted. However, of the fifty articles, only fifteen aligned with the current study. These articles looked at novices attempting to learn about computer programming. The next section summarizes these articles into categories.

Students taking charge of learning.

Two studies attempted to measure the effectiveness of students taking charge of their learning. The first study looked at the effects of PAIRS in improving student competence in computer science while lowering their dependence on teacher assistance. The next study used qualitative methods and produced anecdotal findings that showed that participants learning collaboratively learned equally well as those students not participating in the collaborative activity (Matzko & Davis, 2006).

Student perceptions on pairing.

Another set of studies looked at student perceptions in regards to paired programming. Using quantitative methods, the first study found positive student perceptions when using PAIRS as a pedagogical intervention (Sherrell & Robertson, 2006). The second study also measured student perceptions on PAIRS. Using quantitative methods, the study concluded that students benefitted from paired programming because their partner could help answer questions (VanDeGrift, 2004).

This study also found that students written accounts of their experience helped them understand programming further.

Distance pairing.

Distanced paired programming was also studied. One study looked at the effectiveness of paired programming when teammates are not physically collocated; they are geographically distributed. Using quantitative methods, it showed that pairs collocated did not achieve statistically significantly better results than distributed teams (Baheti, Williams, Gehringer, & Stotts, 2002). Similarly, another study looked at discollocated pairs. In this study, pairs programmed in JAVA virtually, using groupware technologies. Using quantitative methods, it showed that there was no statistically significant difference between the pair and solo student scores (Zacharis, 2010).

Guidelines for pairing.

One study looked at how students were learning and developed a set of guidelines for pairing. It showed that members with more expert students still work alone to complete their piece of the programming activity. However, novices, because everything is new and thus difficult, would benefit from pairing. The article suggested the following guidelines 1) pair within same course section, 2) pair by skill level, 3) make sections mandatory, 4) create timely assignments, 5) institute an instructor selected coding standard, and 6) create a pairing-oriented culture, and the most important component to consider is team mate schedules (Bevan, Werner, & McDowell, 2003).

Transitioning pairs to groups.

One study, using mixed methods discussed findings from the results of transitioning pairs to groups. Groups consisted of at least four members. In the group environment, students organized group work themselves. The study found that groups needed more time to complete the task; however, they liked working in groups, particular with the game aspect of the programming activity (Pastel, 2006).

One report looked at peer collaboration within a personal software process (PSP). Although findings showed increased quality in the programming product and student enjoyment, students followed a defined process for programming (Williams, 2000). Finally, one report called into question the benefits of "small" group work. This report discussed five attributes of collaborative learning. For attribute labeled "small group learning", the paper concluded that it was not clear that group learning is any better than randomized sampling (Preston, 2005).

Thus, all but two of the fifteen articles found benefits of paired-programming. Additionally, knowledge transference was not scaffolded in a discovery-based learning environment. Furthermore, although students may have been learning new computer programming material, they were all computer science majors, except for one study. Finally, all studies were set in a four-year educational institution, not a community college setting as the study setting.

Computer Programming: Traditional

One of the earliest high-level programming languages was FORTAN. FORTRAN stands for Formula Translation and is a language designed for numerical

applications (Dale & Lewis, 2007) and thus was used heavily in scientific applications (Shelly, Cashman, Gunter, & Gunter, 2004). Programming in FORTRAN, similar to all traditional programming environments, require a deep understanding of programming language syntax. Programming language syntax tends to be rather abstract to the beginner programmer, for they have no apriori knowledge. This abstract syntax may prove challenging for these students (Black, 2009); furthermore, non-computer science majors in an introductory computer course may feel particularly challenged when trying to understand such programming concepts (Postner, 2003).

Some programming languages that developed after FORTRAN were markedly different (Dale & Lewis, 2007). First, they did not necessitate a heavy understanding of mathematics. In addition, they decreased the need for understanding abstract concepts such as syntax, concepts that relied on instructor knowledge transference. For instance, LISP, List Programming, an artificial intelligence programming language, introduced the notion of symbols to aid with programming. Symbolic programming allowed for an understanding of computer programming rules using mostly words and sentences in lieu of textual, mathematical syntax (Harvey & Wright, 1999) as did FORTRAN.

Since the 1990s, programming languages, such as JAVA, have become even more familial, less abstract. These languages have been developed to allow a programmer to program using "objects" and thus termed object-oriented programming languages (Dale & Lewis, 2007). An object is a textual representation in code of a real world object. While object oriented programming provides for a more natural, less mathematically, syntaxed programming environment, it still requires an understanding of JAVA object-

oriented textual programming language syntax. However, due to this more natural feel, less instructor transmitted knowledge transference is necessary.

These later programming languages allowed for a more familial, less abstract programming environment and less teacher, knowledge transference method. Nonetheless, all such programming environments utilized mostly an individualized, teacher knowledge transference method, not the contemporary effective group, discovery-based knowledge transference method, for the most part (Jonassen & Land, 2000; Pierson, 1999). Not only do most computer courses espouse individualized learning pedagogies, this ability to solve problems "independently" is encouraged and valued by computer educators (Hoganson, 2008).

Computer Programming: Alternative Approaches

There are relatively scant studies on computer knowledge transference outside of traditional means. One study looked at the five most prevalent methods in which computer programming is taught (Tutty, Sheard, & Avram, 2008). They were 1) teacher as the isolated authority delivering a subject, 2) teacher as the authority delivering a course, 3) teacher as the facilitator of students' learning, 4) teacher as a facilitator of a learner-centered environment, and 5) teacher as a member of a learning community.

In the first three categories, teachers were expected to transfer knowledge using traditional teacher-led instructional methods. In the fourth category, the expectations were for the teacher to teach students how to learn in a student centered learning approach, however not necessarily in a group type of environment. Finally, the fifth

category dealt with computer group collaboration; however, not among students, among teachers.

Contemporary computer environments are allowing for a more student-centered knowledge transference environment by providing computing environments with learning supports that are more familial and less syntax dependent. In spite of controversy surrounding the impact of a learning aid on learning such as Kozma vs Clark whereby Clark (1994) and Kozma (1994) appear to agree on the importance of technology attributes' effect on cognitive processing. Although Clark feels, due to the controversy surrounding the effectiveness of technology on learning, that research should be refocused away from the technology's effect on learning to the cost effectiveness of such technology in learning. On the other hand, Kozma feels that research needs to continue in support of validating technology attributes on scaffolding cognitive processing.

Instructional strategies include the use of learning aids to scaffold student's understanding of the learning material (Dick, Carey, & Carey, 2005). In fact, knowledge transference via technologically equipped learning aids can scaffold students' understanding of abstract science ideas (Buckley, Kershner, Schindler, Alphonce, & Braswell, 2004; Gilbert, Bulte, & Pilot, 2011; Treagust, Chittleborough, & Mamiala, 2002).

Contemporary programming environments tend to move even further from a heavily syntax-dependent computing environment towards a more natural, familial programming environment and thus allows for an opportunity for a more student-

centered knowledge transference learning environment. Several studies have shown the effectiveness of such learning environments on students understanding and attitudes towards computer programming (D'Arcy, Eastburn, & Bruce, 2009; Smith & Woody, 2000). Two such promising aids are Alice and Multimedia Computation programming.

Alice 3D is a visual programming environment that allows users to interact visually with objects on screen (CMU.EDU, 2007). One study found that while only 23% of survey respondents found Alice better than other programming languages, narrative findings showed that participants thought Alice made understanding fundamental concepts easier (Sykes, 2007). Thus, this study's narrative findings show that this non-syntax, visual programming environment seems to have had a positive impact on students attitudes towards programming.

Another non-syntax, visual programming environment is Media Computation. Media Computation programming allows users to interact with media (i.e. pictures, sound) to learn programming concepts. One research studying the effects of Media Computation on student's attitudes towards taking an advanced Media Computation course showed that 60% would take an advanced programming course if the mediacomputation tool were used. This lesser reliance on syntax, visual programming environment, seems to have positivily influenced these students attitudes towards programming (Guzdial, 2003).

Computer Programming: Robotic

In recent years, a new type of digital 3D learning knowledge transference aid has begun to appear on the educational landscape in the form of robots. Some of these robots

can roam about autonomously and make sounds. Moreover, some allow for "remote controller" programming. Due to their 3D attributes, these robots can provide a programmer with an obvious indication of success or failure of programming attempts by providing not only visual feedback but also tactile and audible feedback. Feedback in this manner can help decrease cognitive processing for novice students (Moreno, 2009).

Educational institutions are starting to understand the effectiveness of dynamic 3D mediums to motivate students to learning (Arango, Altuger, Aziz, Chassapis, & Esche, 2008), not only for learning about computer programming but also in other courses (Kandi, Hirano, Eaton, & Ishiguro, 2004; Zhen-Jia You, 2006). Technology, such as robots, provides teachers an additional avenue to teach those "elusive aspects" of the learning content (Gura & Kathleen, 2007).

However, given the novelty of robots in education, a plethora of research has not been undertaken, although sparse findings show promise (Bratzel, 2005; Wang, 2004). Nonetheless, when a new technology is made available that shows promise on influencing learning, it is important to scrutinize methods that might best expose advantages of such technology (Leonard, 2002). However, debate exists on this point, particularly the Kozma-Clark debate.

Robotics competitions.

Robotics is gaining in popularity particularly in the area of robotic competitions (Waters, 2011). These competitions allow for not only programming robots, but also building them. One robotics competition saw 250,000 student participants in its competition (FIRST, 2011).

Several of these competitions utilize LEGOs to build their robot and LEGO Mindstorms, along with its Edu-NXT software application to program the robot (Lego, 2010). However, first the robot must be designed and constructed by the team. Then one can use a remote controller to command the LEGO robot to carry out tasks.

Using the LEGO visual software application, users write code in a low-syntaxed programming environment using activity blocks. This code subsequently is converted into executable code by the EDU NXT software application, not the programmer. This code is then downloaded to the LEGO-robot via a USB cable. By pressing the run button on the robot's controller, the robot will carry out the programming tasks that have been downloaded to its memory (LEGO, 2010).

Two robotic competitions, FIRST and KISS make utilize LEGOs. FIRST (For inspiration and recognition of science and technology) Robotics Competition encourages students to understand robotics by building their robots using the LEGO MINDSTORMS robot kit that includes robot parts and tools to build the robot.

Alternatively, groups can build their own robot from scratch and then use sophisticated software to program their robots (FLL, 2011). KISS's (Keep It Simple, Stupid) Botball robotics program does not require any specialized tools, although reusable tools are needed to assemble the LEGO robot using an iRobot Create robot base, (KISS, 2009). The KISS programming environment must be downloaded to a computer (KISS, 2009).

Another category of competitions encourages use of a kit or reusable materials and tools to build the robot. The BEST (Boost Engineering Science and Technology) organization provides a way for student groups to build and program a robot using materials such as PVC pipe and its programmable platform called the BRAIN (BEST, 2010).

ROBOFEST is one of the first autonomous robotic competitions and it allows participants to use any robotic kits or any programming language from icon-based graphical programming to text-based programming (ROBOFEST, 2011). VEX Robotics appeals to K-20 students and encourages use of the engineering design process in designing and building a robot with a kit (VEX, 2010). As with all such competitions, this kit can be purchased from VEX. The robot can then be programmed using a version of C programming language, either easyC or ROBOTC (VEX, 2010).

While these robotic competitions may motivate students to participate actively, the researcher's volunteer work in such competitions shows that robotic programming tends to be accomplished by one individual team member (NLN, 2010). Team members tend to divide work up and work cooperatively on the robotic project.

Robosapien™ V2 robot.

This research purposely used a particular robot, the Robosapien™ V2 robot (appendix E). Even though the robot is marketed as a toy and not necessarily to the college market, its capabilities seemed ideal for this study. This robot provides for a non-syntaxed, purely tactile approach to learn about computer programming by providing for a purely tactile programming experience and providing feedback in a purely 3D format.

This 24" tall second generation Robosapien is capable of "autonomous free roam behavior (i.e., it can be programmed to move around the room) and is capable of multiple levels of environmental interaction with humans" including sensing colors and making and sensing sound (Wowwee, 2011).

The Robosapien™ V2 has two programming modes, positional and controller. In controller mode, with the aid of the hand-held controller, the robot could automatically carry out a series of preprogrammed modular tasks with one instructional code – the objective of the lab. In positional mode, with the aid of the hand-held controller, the robot could carry out a series of individual tasks; however, each task required use of the hand-held remote to code each individual step.

The controller mode consists of three programmable modules - main, vision, and sound in which commands can be stored. The main programmable module is where all robot action commences, including execution of the scripts in the vision and sound programmable modules. When the computer has been put into main programming mode, a user can enter code into the main memory by using the robot's handheld controller.

Similarly, when the robot's computer has been put into sound or vision programming mode, by using the hand-held controller, a user can enter code into either sub-module's memory using the handheld controller. When a user completes programming, the program can be stored and subsequently executed. All of this programming activity is occurring without an understanding of computer language syntax and in a total hands-on environment.

Not only does this robot allow for an alternative programming method, it also allows for a purely tactile programming experience using the remote controller. While students need to still go through similar algorithmic thinking processes to come up with a correct set of steps, they can now code foregoing an understanding of syntax.

Since the goal of this research is not to look at how a student might learn about syntax, but more about how a student might work collaboratively to programm a robot to complete a task, this research used this robot as a learning aid.

So far, this chapter has discussed learning theories and computer programming environments. The next section will conclude this chapter by discussing learning considerations for the novice student in a community college setting.

Considerations for the Student

Any learning environment should consider the needs of the student (Dick, Carey, & Carey, 2005). This study was set in a community college setting. This type of system usually requires that entrants only have a high school diploma or be eighteen years of age (Dougherty, 2001). Official theory has held that this "open door" provided a second chance to correct previous educational deficiencies (Bauman, 2006). These institutions, which came in vogue during the fifties, satisfied an urgent societal need for skilled technical workers (Raby & Tarrow, 1996).

Nonetheless, community colleges have assumed an increasingly central role in the nation's education and training system (Kane & Rouse, 1999). To support such training, congressional legislation such as the Carl D. Perkins Act of 2006 provides funds that help

develop the academic and technical skills of students in a community college setting (THOMAS, 2011). Moreover, it is forecasted that skills that can be garnered from such institutions will be in demand over the next seven years (BLS, 2009). The "fastest growth will occur in occupations requiring an associate's degree" (BLS, 2009) and the need will be for computer expertise (BLS, 2009; Combs, 2010). Thus, it appears to be a prime time to pursue at least an associate's degree in a computer related occupation.

Next, students today seem skilled with computerized gadgets. After all, to date, they have used technology to send 250,000 instant messages; and play 10,000 hours of video games (Saunders, 2009). Some of them have become known as "gamers". The average age of a gamer today is around 34 years of age (ESA, 2011). Gamers are players of video games (Wikipedia, 2010). Moreover, gaming is starting to be used for educational purposes (Denner, Werner, Bean, & Campe, 2005).

However, this surface understanding of how to engage with a computer via a gadget may not be sufficient for success in the emerging digitally connected world. A deeper understanding is required (Araujo, Filho, & Losada, 2005). Moreover, computer technology constantly changes. According to Moore's law computing capacity doubles approximately every eighteen months (Webopedia, 2011).

Conclusion

This section has discussed literature bearing weight on the study. It discussed learning theories, with an emphasis on group based, discovery-based learning theories, the study's subject matter, computer programming, and the study's participant, a novice,

non-computer science major at a community college. The next chapter will discuss methods used to collect data for the study.

CHAPTER THREE:

METHODOLOGY

The phenomenon this study sought to investigate was *how* (emphasis added) students engage in group-collaborative computer programming in a discovery-based learning environment. Specifically, the following research questions anchored this study:

Research Question One: What interactions and computing behaviors exist when groups engage in collaborative computer programming in a discovery based learning environment with the aid of a robot?

Research Question Two: After completing the collaborative computer programming activity, what are students' perceptions on their robotic group collaborative computer programming experience?

This study explored how groups interacted and collaborated in a group computer-programming to program a robot to complete a task in a discovery based pedagogical learning environment. And as the literature review discussed, this type of learning environment supports maximal, meaningful, lasting learning.

While the goal of the computer-programming lab was to successfully program a robot, this research did not set out to necessarily measure learning effectiveness or provide for immediate generalizability of its results. The objective of this small case study was to describe how college students, along with group mates in a one-semester

introductory computer course work together to program a robot in a discovery-based learning environment.

This chapter will discuss the rationale for the methods used for this study. Subsequent sections in this chapter will focus on procedures, data collection, and data analysis. This chapter will conclude with a discussion on reliability and validity of the research study.

Qualitative Methodology Overview

In an introductory computing course, during the fall of 2008, the researcher introduced a group robotic computer programming project for the first time. With minimal teacher-led instruction and observation, student groups successfully programmed the robot and anecdotal assessments pointed to learning. However, the researcher wanted to determine how students might engage with group mates by more closely observing their group behavior in a discovery-based learning environment.

Thus, the current study sought to determine how students might learn about computer programming in a group-collaborative, discovery-based learning environment. In addition, the researcher sought to identify and describe some defining characteristics that these novice students brought to their programming experience. Identification of such characteristics may inform educator's further consideration for group collaborative discovery-based learning environment.

Qualitative methods have often been used to for understanding people's behaviors and engagements. One should use qualitative methods when looking to see how people

engage in social activities (Carspecken, 1996; Denzin & Lincoln, 1998). In addition, when doing qualitative research, the researcher should have a close relationship with the subject matter (Denzin & Lincoln, 1998).

One should use qualitative methods when looking to see how people engage in social activities (Carspecken, 1996; Denzin & Lincoln, 1998). In addition, when doing qualitative research, the researcher should have a close relationship with the subject matter (Denzin & Lincoln, 1998). This researcher has over 15 years of computer programming experience in the software industry and over 10 years of teaching computer technology courses in higher education.

Procedures: Setting

This research took place at a suburban community college district in Texas. This district stretches from northwest Houston to the town of Conroe. Novice computer programming students from two sections (N~40) of an introductory computer technology course, taught by the researcher, were invited to participate in the research.

Both sections met on Mondays and Wednesdays during the semester for approximately two and one-half hours on each day. Interactive lectures were held on Mondays and lab training on Wednesdays. Lectures included whole class interactive discussions on computer hardware, software, and the Internet.

Labs consisted of training on the Microsoft business suite. Students were expected to complete course objectives as outlined in the course schedule (appendix A). One course objective was to complete a computer programming project. The goal of the

project was to introduce and expose students to fundamental computer programming concepts by programming a robot.

Procedures: Participants

Students from the two sections who agreed to participate in the study were subsequently divided into small groups of three to four members. This resulted in each course section containing four groups, or eight participant groups. Groups met in a lab-like environment (Figure 1), outside of the classroom, five times over a two-week period.

The lab was set up in a conference room in the college library. The conference room, hereafter referred to as "the lab", contained one long conference room table with twelve chairs around it.

The room was somewhat small and narrow. There were two whiteboards on the walls and no windows. There was one picture on each wall in the lab room. These pictures contained hues of reds, blues, and greens. Although not purposely hung, the hues in these pictures become important when students were trying to program the robot to be responsive to colored objects.

A computer cart tower with an internet-enabled laptop computer and overhead projector was provided for student use. It was positioned in the front left of the room near the entry. One video camera positioned on a tripod was used to videotape lab activity. It was situated near the assistant towards the back of the room. Finally, the robot and its accessories were placed on the lab conference room table. The accessories

consisted of the robot's remote controller, the bowling ball and pins, a measuring tape, and ruler.

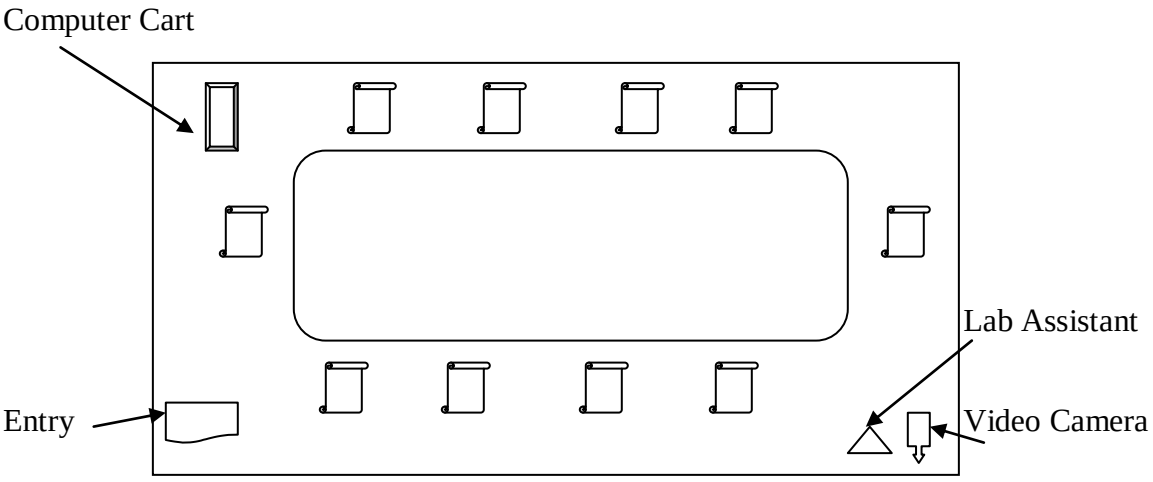


Figure 1. Computer Lab Room

Pre-Lab Initiation Phase

This section will discuss the research study in three phases: Pre-lab initiation phase, Lab phase, and Post-lab phase. The Pre-lab initiation phase will discuss study prerequisites including human subject requirements, assistant preparation, student preparation, and data collection prep. The Lab phase will provide a discussion of how the researcher thought the two-week programming lab would unfold. Finally, the Post-lab phase will discuss data collection and data analysis.

Human subjects.

The researcher sought and received human subject's research approval. Given that the researcher was also the educator, and to minimize researcher/educator/student conflict of interest, IRB mandated the use of a research assistant to facilitate the research

during those times when research data was to be collected; specifically, during those times when students were in the lab.

Assistant preparation.

It is important to consider the observer characteristics (Burstein, Freeman, & Rossi, 1985). After deciding that a former student would be best qualified to serve as an observer/assistant, due to their familiarity with the robotics lab, several former students from the fall, 2008 class were contacted. The chosen assistant and researcher then had an initial meeting two weeks prior to the planned start of the study. The assistant's contract (appendix B) that detailed his responsibilities was reviewed. The assistant's major responsibility was to coordinate lab operations and videotape the lab activity. The assistant had responsibility for lab logistics and robot operation while students were in the lab. However, the assistant was not responsible for teaching or grading the robotic programming unit.

Next, the instructor provided a demonstration of the robot and its programming capabilities. The robot was powered on and the manufacture's built-in demo was initiated by pressing the demo button on the remote controller. Next, several of the robot's capabilities were demonstrated by positionally programming the robot. It had been two years since the assistant had taken the course. The assistant was encouraged to continue becoming reacquainted with the robot before the first lab day.

Table 1 details the researcher and research assistant's responsibilities.

Table 1.

Researcher vs. Assistant Responsibilities

Data and Scope	How	Who	When collecting
Student data sheets (40)	Distribute forms in classroom	Researcher	Pre Lab (beginning of the semester)
Computer project handouts (5)	Distribute in class and/or print from online class website	Researcher	Pre Lab
Research consent forms (40)	Distribute and collect in class	Assistant	Pre Lab
Record the lab activity (8 groups)	Videotape	Assistant	During the lab
Student reflection papers (~40)	Student writes and uploads to online class website	Researcher	Post Lab
Assistant field notes – group summary (8)	Write in field journal	Assistant	During and/or post lab
Document lab activity of all groups (8)	Transcription	Researcher	Post Lab
Perform Interviews	Individualized interviews	Researcher	Post Lab
Transcribe Interviews	Transcription	Researcher	Post Lab

Students could sign and return forms at that time or return them on the first lab day.

Lab Operation

Groups' project activity in the lab was videotaped. The college's Office of Technology Services (OTS) delivered and setup an internet enabled computer cart, projection system, and video camera for each day of the lab. The research assistant ensured proper operation of the robot and video equipment. Once participants entered the lab, video recording began.

At the end of the lab day, OTS picked up the lab equipment and returned it to the OTS offices. Video for that day was downloaded from the video camera's hard drive onto DVD-R tapes and stored. The original video was deleted from the video camera's hard drive. OTS would return these tapes to the research assistant on the next lab day. All data was kept in divisional offices until such time the researcher could view the tapes; after grades were turned in for the semester.

Student preparation.

The week prior to the start of the study, the assistant was introduced to the students along with a discussion of his responsibilities as follows: distribute and collect consent forms, decide which students would belong to which groups, video tape the labs, and run each groups final programming script. Although the class project itself was a required course component, students had an option of having their data excluded from the study, without penalty.

The students were told that the assistant was to provide very little in the way of knowledge transference. Only if students became stuck, and could not move forward, could the assistant offer advice, but only after consulting with the researcher. Then, in

the absence of the instructor, the consent forms were distributed. Students were told that they could sign consent forms now or bring them back later. Table 1 details the researcher and research assistant's responsibilities. Student preparation for the lab activity included four in-class activities.

First, students in the course are expected to participate in group activities starting at the beginning of the semester and throughout. In an attempt to increase students' appreciation for group work, at the beginning of the semester, students participate in a "power of the group" activity. James (2005) proposed that the average of a groups' answer to a problem was usually better than any one individual's answer to that problem.

As an illustration, a jar of beads was passed around the classroom. Students, while sitting at their computer stations and using an online polling site, entered a guess on the number of beads in the jar. The actual number of beads in the jar was 1500. Student guesses ranged from 500 to 3000. No one student guessed 1500. No one student guessed 1750. However, the average of all student guesses was 1750. This group-averaged number was closer to the actual number of beads in the jar than any one student's guess - pointing to the power of the group work.

A lecture on the basic concepts of computer programming was provided prior to the start of the lab via a two and half hour lecture. This lecture did not prescribe how to solve the computer programming problem.

This lecture provided an overview of programming and programming languages, particularly script programming languages. To introduce programming, students engaged

in a simple programming activity. Along with me, they opened a DOS window on their computer and entered a simple DOS command line code: “echo Hello”. After entering the code, students were told that they had just programmed the computer. The syntax of the “echo” command states that whatever follows the echo command will be printed on the screen (Microsoft, 2002). By typing the echo command, followed by the word Hello in DOS, they commanded the computer to carry out a task, display the word “Hello” on the computer’s monitor.

Next, the class engaged in an activity in logic. To get students thinking the logical way computer programmers must think, the instructor had students engage in a seemingly straightforward activity, preparing a peanut butter cracker. Students were instructed to write on a sheet of paper, step-by-step instructions, on how to prepare a peanut-butter-jelly cracker. The instructor then randomly chose three of the students’ instruction sheets.

Carrying out exact instructions, as written by the student, and with the proper ingredients in hand, I attempted to assemble the cracker. One such attempt netted an entire peanut butter jar on top of a single cracker for the instructions stated to “get a cracker and then put peanut butter on it”. It only took a few renditions of this exercise before students seem to grasp the idea of the logical precise steps necessary to accomplish the task.

Finally, students were introduced to the course’s robotic programming project by providing a demonstration of the robot’s capabilities and discussing lab packet handout. First, a demo of the robot’s functionality and capabilities were shown. The

manufacturer's demo was played by pushing the appropriate buttons on the remote handheld controller. Then, individual functionality, such as commanding the robot to bulldoze forward and make a laughing sound was shown using the remote.

Next, students were provided with a lab packet of materials including the robot's owner's manual, lab handouts, sample lab deliverables (see appendix D), and a list of websites where they could view demos of the robot carrying out different tasks. These materials were briefly discussed. Using a feature in Microsoft Word, students were shown how to create a simple programming flowchart, one of the lab deliverables, with the aid of the word processor. They would actually learn word-processing skills in the upcoming week during regular class time when they were not in the programming lab.

The Robosapien™ V2 robot.

This research purposely used, as discussed in the literature review, a particular robot, the Robosapien™ V2 robot (appendix E). This robot is marketed on the manufacturer's website as "combining fluid biomechanical motion with a multi-sensory, interactive humanoid personality. This 24" tall second generation Robosapien is capable of "autonomous free roam behavior (i.e., it can be programmed to move around the room) and is capable of multiple levels of environmental interaction with humans." For example, it can sense colors, movement, and sound.

Even though the robot is marketed as a toy and not necessarily to the college market, its capabilities seemed ideal for this study. This robot provides for a non-syntaxed, purely tactile approach to learn about computer programming. As stated earlier, one of the more difficult aspects of learning about computer programming is to

understand the abstract syntax required to program in traditional programming languages. Thus, understanding of such was concepts were thought to be minimized with the non-syntax programming required of the robot.

The Robosapien™ V2 has two programming modes, positional and controller. In controller mode, with the aid of the hand-held controller, the robot could automatically carry out a series of preprogrammed modular tasks with one instructional code – the objective of the lab. In positional mode, with the aid of the hand-held controller, the robot could carry out a series of individual tasks; however, each task required use of the hand-held remote to code each individual step.

The controller mode consists of three programmable modules - main, vision, and sound in which commands can be stored. The main programmable module is where all robot action commences, including execution of the scripts in the vision and sound programmable modules. When the computer has been put into main programming mode, a user can enter code into the main memory by using the robot's handheld controller.

Similarly, when the robot's computer has been put into sound or vision programming mode, by using the hand-held controller, a user can enter code into either sub-module's memory using the handheld controller. When a user completes programming, the program can be stored and subsequently executed. All of this programming activity is occurring without an understanding of computer language syntax and in a total hands-on environment.

To capture an authentic accounting of how novice students might work together to program a computer, there was an assumption that students had no prior exposure to computer programming or the robot. This assumption was made because this was an introductory community college computers course. For the most part, non-computer science students take this course to fulfill requirements for certification.

However, in case prior exposure surfaced, one of the eight participant groups was reserved for those students. This group was labeled abstainer group. The abstainer group consisted of not only students who had previous experience with the robot or programming but also students who voluntarily opted-out of having their project data included as research data. There were no abstainers in this study.

Finally, there was also an assumption that programming via the robot's handheld remote controller would provide for a simple straightforward familial approach for students to engage in the act of programming. The handheld remote controller is similar to that used in popular gaming environments today. Even though this study took place in a college environment, the average age of the participants was 33.8; the average age of a gamer today is around 34 years of age (ESA, 2011).

In-Lab Phase: Observation Phase

This section will discuss the lab phase of the study. During the lab phase, students actively participated in the lab with their group mates and the assistant videotaped each of the eight groups. This section will provide a discussion on how the researcher thought the daily lab knowledge transfer would occur. During the lab phase, observation research data was collected.

Day one.

On the first day of the project, and in the absence of the educator, the assistant collected the remaining consent forms and finalized group assignments in the classroom. Some groups self-selected, others were assigned. With consent forms in hand, the assistant divided participants into four near-equal groups, labeled A, B, C, D, for each of the two sections. If needed, one group would be designated as abstainers. The abstainer group would consist of those students who opted out of having their lab activity videotaped. The abstainer group would not be divulged to the educator/researcher during the semester. However, there were no abstainers for this study.

Then, the first group along with the assistant, proceeded downstairs to the lab. Subsequent groups went to the lab when previous groups returned to the classroom. The lab groups rotated in lab for 30 minutes at a time. Given approximately five minutes of travel time between the classroom, which is located on the third floor, and the lab room which is located on the first floor of the building, groups were in the labs for approximately twenty-five minutes at a time. The five-minute travel time also gave the assistant time to prepare for the next lab group.

During pre-lab lecture, students had already been instructed to review all lab handouts and the user manual before the lab began. On day one, students would be tinkering with the robot directly or using the hand-held controller to see how they could maneuver the robot around. They also had access to an internet-enabled computer where they could view demos of the robot in action. Additionally, groups might begin

discussion on selecting one of the two options for the lab. Students were presented with two scaffolding questions to help improve knowledge construction as follows:

1) Pick up a ball, throw it and then make a sound, or

2) Bowl, knocking down at least two pins, and then makes a sound.

One of the deliverables for the project was a programming flowchart. When lab groups were not in the lab, they received training on Microsoft Word in the classroom. Word provides symbols that support creation of flow charts. Other lab deliverables included programming scripts and individual reflection papers. Thus, learning Word at this time in the course schedule was purposely designed to align with the class project deliverables.

Day two.

On day two, students would have had a chance to further interact with the robot to become familiar with its functions and capabilities. Groups might now engage in what I term "controller programming", i.e., using the hand-held controller to program the robot. They would start to explore the robot's three programmable areas, the main, vision and sound systems manually with the controller. In order to make the robot carry out one of the options, groups would have to at some point understand that certain tasks must be completed in a sequential order by following a pre-determined algorithm or script. They would start to question how the robot's main memory and sub-memory (vision and sound) works in order to have the robot solve the problem at hand.

Day three.

Not knowing whether students would be able to take advantage of a Saturday lab, I still opened the lab for students to come anytime during a four-hour period to play around with the robot. Three students indicated they had attended.

Day four.

By day four, it was expected that students would be more familiar with controller programming. When the robot is put in controller mode, it can be programmed to carry out a task. This task was carried out by following a step-by-step script. Students would write this script, using Word, and enter the script code into the robot's memory areas (main, vision, and sound) using the controller. This is coding. Once code had been entered and debugged, detailed run instructions could be written for the end user. Their final script should contain three main areas, an algorithm with flowchart, the programming script, and run instruction.

What was hoped is that group members realized that they could not run the sound or vision sub modules unless they had entered code into those modules. The sound and vision modules had to be called in a step by step manner from within the main module. When understanding modular programming, one understands, among other things, that memory areas in a computer can only be used for specific tasks. The code for these tasks can be stored in memory in a segment of code known as a module, for the robot. On day four, students would confirm their run instructions and debug their code as necessary.

Day five.

Groups would demonstrate their final program to the research assistant. To demonstrate their project, the group had to first enter their final code into the robot. One member read out the code while a second member would enter code into the robot using the hand-held controller. After code had been entered, the group would hand the assistant the run instructions for the program and he would carry out those instructions exactly as stated.

Additionally, a peer-evaluation form (appendix F) would be filled out and turned into the assistant. This scored peer-evaluation form would suffice for a student's lab grade. The teacher did not score the lab activity. Groups' programming scripts and individual student's peer-evaluation forms were then stored in the division offices. These papers would be viewed by the researcher at the time of grade assignment. Finally, after lab completion, students would write lab reflection papers about their experiences in the lab. Thus, in summary, students had to turn in the following:

1) Lab paper, which depicted their group's algorithm, flowchart, programming script, and run instructions
2) Peer-evaluation form for each group member
3) Lab reflection papers which would be uploaded to the course website.

The average peer-evaluation scores were used to assign lab grades for each student. Students could upload reflection papers any time after the lab, but before the end of the semester.

Post-Lab Phase: Interview Phase

This section will first discuss how Lab Phase observation videos and transcripts were analyzed to identify participants for the Interview Phase. The purpose of the Interview Phase was to gain a better understanding of why participants were engaging in the programming activity as shown on the video and in addition gain participants' perceptions on their group-collaborative computer programming learning.

The next section will discuss how interviews were carried out and transcribed and how low-level analyses of these interview transcripts were completed. In this section a discussion of how low-level analysis of the first interview necessitated a second interview which resulted in 381 raw codes. Finally, this section will provide a detailed analysis of how these raw codes were analyzed to determine major themes.

During the Observation Phase, the assistant videotaped eight groups during their computer lab activity. After semester grades were turned in and as mandated by IRB, the researcher then viewed the eight videos for the first time, approximately 900 minutes of observation video tape data.

First, videos and transcripts from the Observation Phase were reviewed to determine participants for the Interview Phase. Since the purpose of this study was to describe group effectiveness, groups who were successful at programming the robot were chosen. Next, purposive sampling was used to stratify groups further. Groups who experienced the phenomenon under study (Rudestam & Newton, 2007), i.e., actively engaged in collaborative discovery-based computer programming learning as witnessed by the observation videos, were selected.

As stated prior, this research sought to explore how students collaborated with their group members. However, collaboration requires shared knowledge transference (Barkley, Cross, & Major, 2005). Thus, all group members, not just one student leader had to be engaged and sharing in knowledge transference. Thus in order to be considered a candidate to participate in the Interview Phase, groups had to meet the following criteria: 1) had to be successful in achieving the lab goal and 2) all group members had to be dynamically engaged in knowledge transference over the two-week lab time.

An analysis of the Observation Phase videos and transcripts resulted in the selection of two of the eight Observation Phase groups for participation in the Interview Phase. The final two selected groups were labeled Group A and Group B. Group A consisted of three members, group B four members.

First interviews.

Two months after the end of the lab, members in Group A and Group B were contacted via e-mail and by phone to inquire as to their willingness and availability to participate in the Interview Phase of the study. Although attempts were made to give group interviews, all interviews were eventually conducted individually, due to group member scheduling conflicts. All three members of group A were available for individual interviews and two members from group B were available for a total of five participants for the Post lab Interview Phase.

All interviews were conducted by the researcher and were audio-recorded. The purpose of the interviews was to get an understanding based on participants understanding of why they were engaging with and collaborating with their group mates

as shown on the videos. However, initially, the researcher developed an interview protocol by watching the lab videos. Those times when group members were dynamically involved in programming activities with each other, but additional clarification was needed were captured in the interview protocol.

During the interview, the observation video was played back and at those times where the researcher had noted in the interview protocol, the video was paused for elucidation and elaboration. While the researcher had previously noted those areas to pause the video for questioning, the interviewee could also pause playback at any time to make a comment. This method of interviewing is known as IPR, interpersonal process recall. IPR allows participants to "recall feelings and interpret their behavior" during video playback (Kagan, Krathwohl, & Miler, 1963).

The researcher interviewed the five participants and later transcribed their interviews.

Then, analysis of those transcriptions began. A preliminary analysis of the first interview transcriptions was conducted using low-level coding methods to develop raw codes (Carspecken, 1996). However, some attempts at low-level coding were left open to speculation and/or was unclear and thus needed further investigation. Thus, a second interview took place to delve further into those unclear areas that emerged from the first interview.

Second interviews.

The five interviewees from the first interview were contacted; however, only three members were able to participate in the second interview: two members from group A and one member from group B were interviewed. Fortunately, the clarifications that were needed were from these three members, for the most part. While there was one piece of data needed from the non-participant in Group A, this data was subsequently obtained via an email correspondence between the researcher and the non-second-interview participant.

The second interview took place for the purpose of not only delving further into unclear issues from the first interview, but also to gain the participants' agreement on their first interview transcriptions, in essence providing for member check (Carspecken, 1996). While participant agreement was ascertained with the three second-interview participants, such corroboration was not secured with the other second interview non-participants. The researcher conducted the second interviews and transcribed them.

Initial Data Analysis

Then, initial data analysis of all Interview Phase participants' transcriptions. commenced. In lieu of linguistic analysis, which analyzes each word or phrase in a text to produce findings, a sociological analysis approach was taken to analyze transcribed data (Denzin & Lincoln, 2003) in which all analysis followed open coding and all codes emerged from the data (Corbin & Strauss).

First, over 100 pages transcription documents were read looking for those passages pointing to how participants were actively engaged in the programming activity

either with one another or with the robot. Next, the action sentences in these passages were singled out and became known as raw codes (Carspecken, 1996). There were 381 raw codes resulting from the Interview Phase transcriptions of the study and they were developed very close to the data (Carspecken, 1996, p. 150). The next section will provide a detailed analysis of these raw codes and discuss how they morphed into emergent themes.

Detailed Data Analysis

An Excel document was created with five worksheets, titled by the five Interview Phase participant's pseudonym initials. Each of the five participant's verbatim raw codes were copied and pasted into separate worksheets into an Excel document, into a column titled "Raw Codes". In an adjacent column, titled "Raw Codes Number", each code was numbered. Upon review of the raw codes, some participant's raw codes repeated over the two-week span. Duplicate codes were purposely kept to draw attention to that particular code's frequency of use by that participant.

Next, low level coding began. The first participant's data in the Excel worksheet was reviewed by analyzing the first row of raw code in the Raw Codes column, looking for meaning. A near-verbatim code to this raw meaning code was devised. This meaning code was captured in an adjacent column titled "Low Level Codes". A similar analysis continued for the rest of the raw codes for that participant.

However, some of the raw codes began to share the same low-level code name. These codes were captured in another column titled "Resulting Low Level Themes" and

numbered. As a trace, each raw code number associated with a low-level theme was captured in an adjacent column titled, Raw Code Number Association (appendix H-L).

The next four participants' data were analyzed similarly. If needed, new low level themes emerging from any one participants' data were added to previous participant's codes. What resulted were 15 low level themes across all five participants.

For example, three of Todd's verbatim sentences that were turned into a meaning code were:

"I was trying to figure out if it would even be able to knock it down."

"I was trying to figure out if it would throw the ball at the correct velocity."

"I was trying to figure out if it would throw the ball at the correct velocity."

In all three cases, he was trying to figure out the capabilities of the robot. Thus, these three codes were labeled with one of the 15 low-level codes, "Trying to figure out the capabilities of the robot".

Next, these 15 low level themes were analyzed to see if they could be further collapsed into higher level codes. The first low level theme was assigned a meaning code. This meaning code assignment was based on a belief of what may underlie the interaction or verbatim speech act (Carspecken, 1996). The meaning codes that were assigned were devised based on the researcher's more than 15 years of working in the software industry and ten years of teaching computer courses in higher education.

For instance, “Trying to figure out the capabilities of the robot” was combined with two other low level codes, which were similarly generated, “Robot expectations” and “Lab controller challenges”. These three low level codes were eventually subsumed into one high level emergent theme titled, “Frustration with technology”. What resulted were seven high level themes across all five participants.

The major themes were captured on a separate worksheet in a column titled “Major Themes”. As a trace, each low level theme number associated with the major theme was captured in an adjacent column and titled “Low Level Theme Number Association”.

Chapter Four will discuss the major themes using participants’ verbatim voice. For further analysis, the themes will be juxtaposed with Weinstein and Meyer’s (1986) categorization of cognitive learning strategies and Hackman’s (1990) theory of effective group work. Weinstein and Mayer (1986) stated that the use of cognitive strategies, such as rehearsing, elaborating, organizing, metacognition and modifying, or engaging in affective motivational strategies can scaffold learning of a cognitive subject, like computer science (Smith & Ragan, 1999).

Also, as previously discussed, one of the criteria for participation in the Interview Phase of the study was that a group had to be successful in programming the robot. Hackman (1990) stated that group work helps individual members of the group better understand the material without the need of an academic superstar. He also stated that small group-based learning is effective in just about any course with learners aged 18 to 80.

However, in order to be effective, Hackman described attributes that members in a group need to exhibit as follows:

1) Adequate knowledge and skill to bear on the task
2) Task performance strategies that are appropriate to the work and to the setting in which the work is to be done
3) Motivation to exert sufficient effort to accomplish the task at an acceptable level of performance (as cited in Michaelsen, Knight, & Fink, 2002)"

Reliability and Validity

A concern of any research is reliability and validity of the data and according to Merriam (1998), these concerns can be approached through careful attention to "the way in which the data were collected, analyzed, and interpreted, and the way in which the findings were presented" (p. 2398).

Reliability.

Reliability refers to dependability and consistency. Consistency in the qualitative sense is not the same as in the quantitative sense for qualitative does not engage in repeated trials of some phenomenon. Consistency in qualitative studies can be gained "through coding the raw data in ways so that another person can understand the themes and arrive at similar conclusions (Carspecken, 1996)". Each level of the four-level theme hierarchy was traced to a lower level (appendix N). This hierarchy provides a detailed explanation of how transcribed data morphed from raw data to high level themes. This information was captured by tracing data from high level themes to raw coded data.

Validity.

The notion of trustworthiness is often used in qualitative methods in place of validity. This study employed the following measures to increase the trustworthiness of the findings:

1. Member checks via participants
2. Peer debriefing to check possible biases of the researcher's interpretation of data findings.

Multiple sources of data were used including videotaped observations, interview transcriptions, reflection papers, and assistant field notes to help answer research questions for this case study (Stake, 1995). Member checks were completed by having interviewees review their transcribed interviews for accuracy. However, only three of the five Interview Phase participants engaged in member check.

Researcher biases were checked via three methods. First, colleagues analyzed the interview protocol strategy and modifications were added as suggested. For instance, instance of stopping playback only at those areas the researcher sought additional clarification, a colleague also suggested that participants stop playback to make comments at any time.

Next selection of Interview Phase participants was checked against the research assistant's assessment of viable Interview Phase participants. Observation Phase participants were selected to participate in the Interview Phase based on two criteria, dynamic engagement of group members and group success. The researcher felt two

groups met the criteria. While the research assistant felt that four of the eight groups were successful, two of those four groups did not have members who were dynamically engages with their group mates throughout the two-week timeframe in the lab.

Finally, a peer debriefer was hired (appendix C) to develop a set of codes and themes based on transcribed interviews. The attempt here was to provide an additional method to verify the researcher's themes, to increase validity by checking for any researcher bias (Carspecken, 1996). This study resulted in eight major themes. However, the peer debriefer's analysis resulted in eleven emergent themes (see chapter three). There was a direct alignment of eight of the peer debriefer's themes with the study's themes. The remaining three peer debriefer themes were coded as lower level themes within two of the studies higher-level, major themes.

Specifically, four of the debriefer's Themes 1, 2, 10, and 11 confirmed with the researcher's Themes 1 and 2, which dealt with frustration. Another debriefer theme, Theme 9, dealt with "going beyond" which confirmed researcher's Theme 6. Two of the debriefer's Themes 3 and 4 aligned with researcher's Theme 2, "programming frustrations". Finally, Theme 6 "needed the manual" confirmed one of the study's low level themes.

On the other hand, three of the debriefer's themes did not confirm this study's major themes. The debriefer's Themes 7 and 8, although labeled major, was coded as a lower level code and subsumed in Theme 5 of the study, signs of adversity. The debriefer's major Theme 5 discussed computing behaviors and computer language. However, this was expressed as a low level code in the study's major Theme 5,

"frustrations with programming". A summary of the themes emerging from the debriefer's analysis are shown in appendix Q.

Conclusion

In conclusion, chapter three has discussed the methods and procedures used to collect and analyze study data. Over 900 minutes of video were observed and over 100 transcription pages were analyzed which resulted in 7 emergent themes. The next chapter, Chapter Four, will discuss these major themes using participants' verbatim voice. It will provide for further analysis of major themes by juxtaposing major themes with Weinstein and Meyer's (1986) categorization of cognitive learning strategies and Hackman's (1990) theory of effective group work.

CHAPTER FOUR:

FINDINGS

The phenomenon this study sought to investigate was *how* students engage in group-collaborative computer programming in a discovery-based learning environment. Specifically, the following research questions anchored this study:

Question One: What interactions and computing behaviors exist when groups engage in collaborative computer programming in a discovery based learning environment with the aid of a robot?

Research Question Two: After completing the collaborative computer programming activity, what are students' perceptions on their robotic group collaborative computer programming experience?

This chapter will discuss the emergent themes using participants' verbatim voice. Note: For clarification, parenthetical material was added in italics. For further analysis, as discussed in chapter three, the themes will be juxtaposed with Weinstein and Meyer's (1986) categorization of cognitive learning strategies and Hackman's (1990) theory of effective group work. An analysis of how Weinstein's cognitive strategies and Hackman's group attributes manifested in the study's major themes will be discussed.

Participant Background Information

First, a discussion of participant demographics and background information. At the beginning of the semester, all students in the course completed a student data sheet

(Appendix O). Among other things, this data sheet was used to gather contact information to contact participants for follow-up interviews. In addition, students indicated their computer experience by a single indicator, either none, some, or lots. Additionally, before the individual interviews, additional participant background information was captured on the demographic background information sheet (See Appendix P).

Group A

Group A consisted of three members. All members participated in both the observation and Interview Phases of the study.

First, Kathy was an African-American female aged, 47-55 years old. On her student data sheet she indicated that she had "some" experience with computers and "none what so ever" with programming; for robotics' experience, "just simply watched them on T.V. or heard about on the news".

Tanya was a Caucasian female aged, 40-46 years old. On her student data sheet she indicated that she had "lots of" experience with computers; her interview background sheet indicated that she had "no" programming or robotic experience.

Lastly, Todd was a Caucasian male, aged 18-20 years old. On his student data sheet, he indicated that he had "lots" of experience with computers. His interview background sheet indicated that his experience with computer programming was with "First LEGO League in 5th grade". He also indicated that he had experience with computers and robots because he was a member of computer and robotic teams.

Group B

Group B consisted of four members. All four members participated in the Observation Phase; however, only two were subsequently interviewed and thus only these two participants' background data will be discussed.

First, Anna was a Hispanic female, aged 40-46 years old. Her student data sheet indicated that she had "some" experience with computers. Her interview background sheet indicated that she had "no" experience with computer programming and her only experience with robots was in "science museums"

Next, Elaine was a Caucasian female aged 18-20 years old. Her student data sheet indicated that she had "lots" of experience with computers. For experience with programming, she indicated "none personally". Her direct experience with robots was via a family member who had a non-programmable version of the robot, not the robot used in this study.

Themes Analysis

The open coding process, as discussed in chapter three, netted 381 raw codes that were analyzed and collapsed into 98 low level codes. A further analysis of these low-level codes netted seven hierarchical themes as follows:

1. Frustrations with Technology
2. Frustrations with Programming
3. Prematurely celebrating success
4. Elaborating lab objectives

5. Trying to determine how to program the robot
6. Groups members dealing with incompatible working styles
7. Assuming group mates to have programming/robotic skills that they did not have

However, Theme 5, which dealt with programming the robot, was later subsumed into Theme 2. Additionally, Themes 6 and 7 manifested while group mates were trying to construct knowledge collaboratively. Thus, a higher theme titled "constructing knowledge collaboratively" subsumed Themes 6 and 7. The following resulting themes will be discussed:

1. Frustrations with Technology
2. Frustrations with Programming
3. Prematurely celebrating success
4. Elaborating the lab objectives
5. Dealing with circumstances from attempting to construct knowledge collaboratively.

Frustrations with technology

First, Theme 1 dealt with frustrations with trying to understand and use novel technology, the robot and its remote controller. Participants in this study had no prior relevant experience with robots. Most of the frustration students exhibited with the robotic technology was due to unfamiliarity with the robot's capabilities. For instance, Kathy, from Group A, stated she was not "…familiar with what it would do. Not ever having any experience of them other than watching them on TV".

Anne from Group B stated that even though she understood the material in the user's manual, and she had watched online videos of the robot in action, she still was unsure about the robot's capabilities. She exclaimed, "But what he can do in the real life. What does it means when he throws a ball. Like how does he throw the ball? What is, what are his capabilities?"

It became more complicated when participants not only lacked an understanding of the robot's capabilities but felt that understanding such capabilities would necessitate understanding who initially programmed the robot. For instance, Kathy stated that the robot was probably doing what it was doing because "…it depends on the person's personality *(the person that originally programmed the robot)* as to what this robot can do. Whose to know?"

There was also frustration surrounding use of the robot's handheld controller. All coding of the robot was facilitated with a handheld remote controller and its multi-faceted controls and buttons. While the remote controller may have proven awkward to use, hand fatigue may have also resulted from participants trying to program the robot to complete some task of which they did not fully understand how to undertake. Thus, a fair amount of trial and error, hand-held coding and debugging with the remote controller resulted.

Members from Group A indicated their frustration with the controller as follows: Tanya indicated the handheld remote was "…a little bit tedious initially" and Todd exclaimed, "We were trying to work with the remote which I believe proved to be pretty complicated. Kathy opted not to use the controller during the lab. She said that her

"gamer" group mates had more experience and that she would let them do it; however, she also acknowledged "…I guess I could see that they *(her group mates)* were getting frustrated with the robot controller. You have to do this, do that, press this".

One reason Group A took an alternative approach to programming the robot, in addition to wanting to have a more creative lab experience, was to forego use of the robot's controller. Kathy stated, "He *(Todd)* wanted to use Wii controllers so that we wouldn't have to F1 this, and shift this *(enter commands using the multi-faceted buttons on handheld remote)"*.

Elaine, from Group B alluded to the awkwardness of the remote controller when she explained that at one point she and/or a group mate must have accidently hit something on the controller because the robot starting doing something unexpectedly. She said, "I think he *(group mate)* flicked something, or I might have flicked it. All of a sudden, it started walking backwards". In a somewhat relieved way, on day two of the lab, Elaine, who had used the controller singularly on day one, wanted to give up use the controller. She stated, "Someone can use it, I had already used it on day one".

Thus, members from both groups experienced fatigue which led to frustration while using the handheld remote controller and from trying to understand the capabilities of the robot. Most of this frustration exhibited the first two days.

Hackman's group attribute 1, states that group members should have adequate knowledge, and skill to bear on the task (Michaelsen, Knight, & Fink, 2002). It is obvious from the discussion that members did not have adequate knowledge or skill to

use the remote. However, more importantly, in a GCD, it is important to use learning artifacts should support student learning, not detract from it (F. Fischer, personal communication, April 10, 2011; Bruner, 1961; Duffy & Cunningham, 1996; Howland & Jonassen, 2003).

Even though members expressed frustration with understanding the robots capabilities and using the remote controller, they continued to work on the lab assignment not only at an acceptable level, according to lab guidelines, but with motivation. This motivation may have been fueled by participants desire to continue engaging with the humanoid-like robot. All participants seemed to be intrigued by the robot and took delight in its various humanoid-like sounds, movements, and auto-roaming behavior. While there was a fair amount of frustration, there was also a fair amount of laughs. Weinstein states that understanding a cognitive subject can be scaffolded by engaging with an affective motivational strategy, such as the humanoid-robot appears to have provided.

Frustrations with programming

Once again, computer programming was new to these participants. Furthermore, they were learning without the benefit of expert knowledge. When constructing knowledge in a discovery environment, the learner will most likely struggle. However, this is learning (Bruner, 1961; Rodrigo & Baker, 2009; Duffy & Cunningham, 1996). Moreover, this form of learning will have meaning, and meaningful learning subsequently leads to lasting knowledge (Bell, 2003; Barkley, Cross, & Major, 2005; Denzin & Lincoln, 1998).

Programming frustration mounted over the five-day period with maximum frustration occurring on lab Day Two. There was considerable frustration experienced by both groups from debugging and revising the programming script, a normal part of computer programming. However, these novices were not aware that debugging was a natural part of coding. Some participants approached these "trial and error" episodes as a necessary part of the lab, while others became frustrated with doing repeated trials that were not effective.

For instance, when asked what trial and error meant to her, Tanya (Group A) stated, "trial and error was playing with it, making it talk, doing different things and see…it was important to do many trials and see what errors we got out". On the other hand, her group mate, Kathy felt that they might fail the lab because they were doing so many trials, without success. She exclaimed, "We are going to fail… trial and error meant you would try different things each time, not the same thing…going through it day by day…I thought it was stupidity".

In addition, when asked if she thought trial and error was frustrating, she exclaimed, "Frustration...Yeah. Wow, we are going to do this all day long. As I see it…Let's do something different…it seems like we're wasting…and nothing was working".

Elaine (Group B), like Tanya, felt trial and error was not necessarily a waste of time, but a part of the process: "We had entered all these commands and it didn't do anything…We keep flipping…We knew by trial and error that if it saw the ball, it wanted to bowl. A process of elimination"

Frustration also centered around trying to understand modular programming; entering the appropriate code into the robot's appropriate programmable module (main, vision or sound). Students had trouble understanding how to enter the code in the correct module, what order to enter the code in each module, and understanding how code was subsequently stored and executed. Understanding such programming concepts within a two-week span was challenging at the least. Elaine's depiction of programming frustrations sums up experience for all group members:

"We didn't realize that we didn't actually store any programs because we weren't opening each individual visual and audio. I only entered one command. I didn't know whether you had to do it four times to clear all four commands or if. We're just trying to break it up into both sound and vision. We went in there realizing you know that it had to be three separate entities. Didn't realize you had to open that command. We kept putting commands in any way not knowing that it wasn't. Figuring out that it had to be three separate things. We really weren't sure whether or not we really had to break them up"

Trying to split it up the code into modules was the most challenging: "We weren't really sure how to how to break it up. We thought we had to open the main program and enter everything in there. We didn't realize that you could do them separately, without having to be in the [other groups] they had stored their program. So that if you don't clear it then you can do the two small things and then open up bigger thing.

However, upon reflection while viewing the observation videotapes, Anne (Group B) thought writing down what they had done would have been helpful. She stated:

"And everything we had did, we don't know what we had done, because he had. We were just trial and error, trial and error. At the end of this one, we knew what we were definitely wrong. We weren't writing it down so that we would know what we were doing wrong. We kept pushing and pushing and pushing. Frustration elevating because we were doing, doing, doing, doing, doing. That would have been a really big, to see what we had done, because we did it. And we put it in and we was not erasing the previous one. It did what we wanted to but, it started so many things and it was frustrating. Clear the program, start the program, then end the program. We're doing the same mistake that you have to put the vision, then sound, then. What are we doing wrong?"

In summary, participants' frustrations from trying to program the robot stemmed from them trying to understand modular programming when they did not understand concepts such as storing and execution and subsequently ended up in repeated trials and errors.

Hackman's attribute 1 states that group members should have adequate knowledge and skill to bear on the task. Once again, this lack of knowledge was expected in this discovery-based environment. Additionally, Hackman's attribute 2 states that task performance strategies should be appropriate to the work. Group A's lab strategy was not in line with the lab objectives. As group member Kathy stated, it was "above and beyond" the expectations of the lab.

Weinstein suggested use of organizing as a cognitive strategy when learning cognitive subject matter. As Anne (Group B) implied, if group members had been keeping track of trials and errors they may have been able to move forward with less trial and error and subsequently less frustration.

In summary, even though Hackman's attribute 1 states that group members must present with adequate knowledge and skill to bear on the task, due to this lack of adequate knowledge, both groups sought knowledge and where successful in the end. Partially because Hackman's attribute 3 states that members must have the motivation to exert sufficient effort to accomplish the task and it appears both groups were motivated, sought additional knowledge and were successful.

Celebrated premature success

On the first day of the lab, both groups thought they had successfully completed the lab objective. Both groups experienced premature programming success. This premature programming success most likely stemmed from novices not understanding what they have to do to be successful (Pierson, 1999) to program the robot. Both groups had successfully programmed the robot in positional mode, not controller mode programming. However, controller mode programming was the objective of the lab. Tanya, from Group A turns to the assistant and asks, "Are we done"? Similarly, Anne from Group B turns to the assistant and asks, "Is it really that easy"? Elaine, upon reflection stated that at this point they were thinking, "it was a lot easier than what we had thought"

Moreover, even though both groups had positionally-programmed the robot to complete a task on day one and even though the lab assistant informed both groups that they had not programmed correctly, on day two, both groups resorted to positionally-programming the robot. It appears that since they were successful in commanding the robot to complete a task positionally on day one, they continued in this vane, despite being corrected by the assistant.

Upon reflection, Anne explained why groups may have continued in this vane:

"We stuck to that. And then we came in the second time and we went ahead and put it in the main. I mean like the first session we were able to play with the robot we were able to do it. It's easy and it's going to work…got us really excited when he did everything. We didn't change it; we had already made it through. Like instead of making some other things and doing some of the trials we were. We kept sticking to the script. And we didn't even know what we were doing wrong. Everyone was really excited we thought it was very easy and you have".

Group members did not question success within the group for group members did not understand the requirements for success. And since they did not understand the requirements for success they could not think about if their celebration of success was logical. Metacognition is thinking about thinking. Weinstein stated that metacognition and modifying strategies can help scaffold cognitive learning, such as computer programming. While both groups were engaged in metacognition somewhat, for they did question the assistant on their thoughts of success, they did not question within the group because they did fully understand what it meant to be successful.

Going beyond lab objectives

Groups could choose to work on one of two lab assignment options, as described in the lab handout. The programming options, lab materials and handouts were purposely devised by the course instructor to facilitate group work in a short amount of time. One of the lab deliverables was a programming script. To facilitate their understanding of this deliverable, participants were provided with an actual template of the final deliverable code. The only parts missing from the template were five lines of code. Participants had to decide which additional actions or tasks they wanted the robot to complete and add those lines of code to the template.

While Group A initially chose one of the lab options, they subsequently decided to pursue an alternative, yet challenging approach. Their programming activity necessitated understanding advanced robotic capabilities such as precise movements and how to use the robot's extended vision capabilities. They spent a fair amount of time the first two of the five days taking measurements and trying to get the robot to recognize

color swatches that they had gotten from the local home supply store. Their goal was to have the robot walk up to a tower, recognize a color, and then knock over the tower. By day four they realized they were not being successful and then opted to return to their initial lab choice, in which they were eventually successful.

Their creative adventure stemmed partly from an assumption about their skills with robotic programming and the remote controller. For instance, Tanya emphatically stated, "Well you know, I'm a gamer". Todd stated, "I was coming from a background with the impression that I could tell the robot exactly what I wanted it to do within its physical parameters". Subsequently, they took on the more challenging, although creative solution. Todd acknowledged that only "after understanding the limitations of the robot", on the last lab day, "that they were not going to able to use it" *(their design)*. He stated, "We ended up going back and trying to figure out a way to make it do what we had originally".

Group B chose to solve one of the carefully designed lab options. However, despite this fact, Group B still encountered frustration, similar to Group A.

Constructing Computer Programming Knowledge Collaboratively

In trying to construct knowledge collaboratively, groups had to deal with their team mates working styles and they had to figure out how to program the robot, they had to discover knowledge.

First, for the most part, all members in both groups worked well together. However, there were some signs of adversity when constructing knowledge

collaboratively. For instance, Kathy (Group A) did not agree with her team mates programming approach. Kathy felt that her group mates' approach to programming was ineffective. She thought they should be trying new things, not repeating the same errors.

Kathy explained her frustration by stating that her group was trying to "….go above and beyond" without understanding the robot's capabilities. "We still were not familiar with the actual robot and what it could do. …what his limitations were and how far we can push him". She also said she understood her group mate's tactic of using the alternative approach to programming the robot, however, "…how could they look at alternatives when they did not really understand the capabilities".

She indicated that she did not voice her opinion on this matter because after all, one of her group mates, Todd, was a "techie" and the other Tanya was a "gamer" and they looked like they knew what they were doing. Moreover, she looked to her group mates for understanding. While she acknowledged her group mates gaming skills, she also stated, "Playing a game and programming a robot are different".

She also acknowledged their desire to be creative by stating "They always have new ideas. "They are aggressive" and "You know, far be it for me to stop them." Regarding her group mate Tanya, Kathy stated she is "…aggressive, " but "None of us really, you know, we really don't mind because that's what she likes." Later she stated, "Well she was a little bit more aggressive. She kinda took over a leadership role. We just let her do it. You know, I played along". Then she goes on to justify Tanya's behavior by saying, "She just wanted to get it done".

Todd felt the robot should have been less human-like because that may have helped his group mates see the robot as a thing. He stated "…they should not have treated it like it was human" as implied by Kathy's statement, "He's *(the robot)* just like humans, kind of. You know, the closer it is, the more identifiable it. This is something human." He felt that this personifying interfered with programming because he felt that his group mates were expecting the robot to have certain automatic human capabilities that it did not have. He goes on to state that it would have been better if it looked like an "RC car", then maybe they would not expect so much humanness out of it.

Also in Group A, Tanya discussed with hesitation Todd's programming approach. Even though she felt he merely "had a different approach" and "that was his right", it was different from her approach. She stated,

> "He wanted to just take a few commands and program it, but in his way, not according to the manual. Again, it was just a different way of doing things. We are on different levels. But, I wanted to program by reading the manual, following the instructions. I am one that go according to the instructions"

All members in Group B worked working styles meshed on day one. However, a new member joined the team on day two. Anne stated that he "just came in thinking that he knew what he was going to have to do". Anne said she expected the new member would be a great addition because "he came in playing with the robot and the controller as if he was experienced". However, by displaying this level of comfort, she assumed that he "also knew how to program it, how to divide up the code into the correct modules".

However, she reconsidered. He began playing with the robot and its controller concurrently while they were trying to program it. Thus, when the robot did not respond as they expected, they were not sure if it was their code or something the new member had inadvertently done. Anne stated, "We don't know what was wrong, if it was the code or was Doug doing something".

When Doug joined, Elaine implied that they would not have time to get him caught up to where they were. She stated, "We kinda skipped all the basics with him and just said, you know, look on". However, later on they did try to include him on the project. She goes on to say that, "we tried to include him, you now, to get him involved".

Secondly, in trying to understand computer programming, group members consulted with their team mates, who were novices, viewed videos of robot capabilities online, which did not provide prescriptive information, read the owner's manual, and tinkered with the robot and its controller, haphazardly. They also sought help from the lab assistant; however, such help was not forthcoming. The lab assistant was instructed to only give guidance when groups became stuck and could not move forward. Finally, they had access to an online discussion board; however, no one took advantage of this opportunity.

Out of all such knowledge seeking avenues a member from each group specifically expressed their appreciation for having access to a hard copy of the owner's manual. Tanya, from Group A, stated she "needed that manual, explored a little bit on YouTube, and learned reading through the instructions through the manual". And even though her group mates learned about the alternative approach online, she stated that for

the most part she was “going by the actual instructions in the manual” instead of trying to figure out how to program the robot by looking at the videos. She so appreciated having access to the manual that she goes on to state, “The manual, I cannot express enough how much that was appreciated”.

Also, Elaine, from Group B, stated the importance of the manuals. She said that she read the instruction manual and watched YouTube videos. She also exclaimed, “We would've been completely lost without those manuals”.

Finally, a pivotal point of knowledge transference occurred for Group B due to one member’s work. After the second lab day, Anne stated she went home and figured out how they must divide up the code. She said she had figured out that the coding could be done in “one of three options”. She then attended the Saturday lab to try out her options. One of her options worked. When she reported to the group and shared her understanding, they all seemed to grasp the concepts as well and seemed appreciative of her additional efforts.

While Hackman states that groups do not necessarily need a superstar to do super work, Anne provided the missing link that her group needed to be successful. She later shared this knowledge with her group such that not only did she gain knowledge, so did her group mates.

In summary, while both groups worked well together throughout the two-week span, some members wanted other members to work differently. In addition, groups had access to various mediums to gain knowledge, however a hardcopy of the user’s guide

proved most effective. Finally, one member in one group provided knowledge transference by working alone. Nonetheless, the group benefited once she shared her new knowledge. Hackman stated that highly functioning group members would engage in task performance strategies that are appropriate to the work and the setting. Allowing group members to work within their natural learning style proved beneficial, as well as having access to a variety of learning material.

To conclude this chapter, a summary of findings based on emergent themes as explained by this eclectic mix of Hackman's group attribute theory, Weinstein's cognitive strategies, and learning theories are provided in Table 2.

Table 2

Emergent Themes vs. Theory

Emergent Themes	**Corroborate with: Hackman, Weinstein Mayer, Learning Theories**
1 – Frustrations with Technology Frustrations increased in attempts to understand the robot capabilities and in using the handheld remote. Despite these frustrations, members not only continued the lab exercise, but with enthusiasm.	Learning Theory: When constructing knowledge in a discovery environment, the learner will most likely struggle Hackman: Group members should have adequate knowledge, and skill to bear on the task. Weinstein: Understanding a cognitive subject, such as computer programming, can be scaffolded by using an affective motivational strategy.
2 – Frustrations with Programming In both groups, while attempting to understand modular programming, frustrations mounted because of the lack of knowledge, all the while being aware of a time limitation. This lack of knowledge subsequently led to repeated trials. Additionally, one group went "above and beyond" the requirements of the lab activity by choosing a task performance strategy that was outside the scope of the lab goals.	Learning Theory: When constructing knowledge in a discovery environment, the learner will most likely struggle Hackman: Group members should have adequate knowledge, and skill to bear on the task; Hackman: Groups' task performance strategies should be appropriate to the work. Hackman: Group members should have motivation to exert sufficient effort to accomplish the task at an acceptable level of performance. Weinstein: Elaboration can be used as a cognitive

Group members lacked sufficient knowledge. This motivated them to exert the effort to seek additional knowledge. Group members sought additional knowledge in different ways; from reading the user's guide, viewing videos online, engaging in discourse with teammates or assistant, or tinkering with robot. More importantly, group members assumed group mates had certain knowledge.	strategy in learning about computer programming. Weinstein: Using organizing as a strategy helps with understanding cognitive material.
3 – Premature Celebration Because groups did not have an adequate understanding of what success looked like, both thought they had completed the lab objective on the first day, when in fact they had not.	Hackman: Members need adequate knowledge. Weinstein: Metacognition and modifying can be used as a cognitive strategy. Metacognition is thinking about thinking.
4 – Going Beyond Lab Objectives Both groups wanted to go beyond the stated lab objective at some point. Group A was motivated by having the ability to be creative from the onset, and Group B motivated after they had been successful in programming the robot.	Hackman: Highly functioning group members will engage in task performance strategies that are not only appropriate to the work and the setting but also may go beyond.
5 – Constructing Collaborative Knowledge	Hackman: Highly functioning group members would engage in task performance strategies that are appropriate to the work and the setting.

While both groups worked well together throughout the two-week span, some members wanted other members to work differently. Furthermore, some members expected their team mates to be the "super star" and to have certain robot and/or programming skills when in fact they did not.	Hackman: Groups do not need a superstar

Research Question Two: Coding and Themes Analysis

To address Research Question Two, participant reflection papers were analyzed for their overall perception of the programming activity. However, only four of the five participants uploaded a reflection paper. The major theme emerging from the reflection papers was that although frustrating at times, participants enjoyed collaborative computer programming. This is important because satisfaction in learning subsequently affects learning outcome (Lo, 2010).

For instance, Todd from Group A stated, "I enjoyed working with my class mates and thought they were extremely helpful in the process of researching and working with the robot." Similarly, Anne from Group B stated "…I believe the work group is a key element as everybody´s ideas enrich the experience". Finally, Kathy from Group A expressed her satisfaction with group work by stated that "Working with my group mates was fun because we worked well together".

One in each group also indicated the need for more time. For instance Todd implied that his group could have used more time but unfortunately their "Group time

was shortened by 5 minutes each time they were able to visit the lab." Anne stated, "The improvements I would make would be to extend the time to interact with the robot as the time pressure ends up in frustration and simple mistakes you would not do if there were more time." Finally four of the five participants at some time during the study expressed their appreciation for having access to a hardcopy of the robot's Users Guide.

Tanya summed up the quandary of discovery-based learning when she stated, "If you don't know what it is that you can accomplish you will never be able to accomplish more".

Conclusion

In conclusion, chapter four has provided background information for each Interview Phase participant and presented the emergent major themes resulting from the study and juxtaposed them with Hackman (1990), Weinstein and Mayer (1986). To address affordances of learning in such an environment, Chapter 5 will suggest recommendations to mitigate negative behaviors as discussed in the findings of this study to help maximize learning outcomes in a GCD learning environment.

CHAPTER FIVE:

CONCLUSION

The phenomenon this study sought to investigate was *how* students engage in group-collaborative computer programming in a discovery-based (GCD) learning environment. Specifically, the following research questions anchored this study:

Research Question One: What interactions and computing behaviors exist when groups engage in collaborative computer programming in a discovery based learning environment with the aid of a robot?

Research Question Two: After completing the collaborative computer programming activity, what are students' perceptions on their robotic group collaborative computer programming experience?

The findings from chapter four showed some defining behavioral characteristics of how students collaborated in computer programming in a discovery based learning environment including, (1) frustrations from engaging with the learning aids, the robot and its remote controller, (2) frustrations from trying to understand the subject matter, computer programming, particularly modular programming, (3) prematurely celebrating success because they did not know what success really looked like, (4) not staying on course of the stated lab objective because they wanted to be more creative, to be challenged, and (5) dealing with adversity in team mates' work ethic while constructing computer programming knowledge collaboratively.

Given these behavioral findings, this chapter will offer recommendations that might mitigate these behaviors and hopefully enhance the benefits of learning in a GCD environment.

Minimizing Frustrations with technology

Frustrations from trying to understand the robot's capabilities was to be expected, for it was novel learning aid of which participants had no apriori knowledge. However, participants had been provided pre-lab study materials to engender familiarity, including the robot's owner's guide and links to online videos highlighting the robot's capabilities and functionality.

Additionally, the researcher assumed that working with the robot's remote controller would be a benefit for this "gamer" generation of student. However, unfortunately, members from both groups experienced fatigue while repeatedly using the handheld remote controller to program the robot. This fatigue was due to using a remote controller meant for play with a robot, not for repeated computer programming debugging.

Teachers are an integral part of a GCD environment. However, teacher preparation in the design of such a course requires not only a certain amount of creativity (Lo, 2010) but also time to consider the technological/pedagogical dynamic (Harris & Hofer, 2011; Pierson, 1999) of integrating technology into a discovery-based learning environment so as to effectively impact the learning experiences of the students.

Thus, the following recommendations are suggested to help minimize frustrations with technology:

1. Although participants had been given robotic learning material and handouts, do not assume that students would have reviewed this material before lab and that they understand the functionality and capabilities of the robot.
    a. Before lab initiation, give a lab packet test to assess if students' understanding of the lab handouts.
2. Because students became frustrated with the technology, consider the affective impact the learning aid might have on the student.
    a. Before use of a novel learning aid, have students complete a survey that points to any biases, concerns or possible satisfactions they may have with engaging with the robot.
3. When working in a computing lab environment, not only should ample time be allotted for programming (Attarzadeh, 2006), additional time should be allotted for students to struggle with novel learning aids. Pilot new aids to ascertain a sufficient time.
4. Do not assume that a popular artifact, such as a gaming remote controller, will easily apply to an educational environment. Determine how students might engage with learning aids.
    a. Before lab initiation, in addition to demoing the capabilities of the robot have each student physically handle the robot and the remote

controller. Then on a scale of 1 to 10, have them rate the ergonomics and ease-of-use of the particular aid.

5. Additionally, to support educators development of a dynamically technology rich learning environment, utilization of several components of the TPACK (Technology, Pedagogy, and Content Knowledge) framework (Mishra & Koehler, 2006; Pierson, 1999; Harris & Hofer, 2011), are suggested as follows:
    a. Sufficient time to integrate new technology into their pedagogical strategy.
    b. Availability of experts to guide and assist knowledge transference.
    c. If available, exemplary models.

Minimizing Frustrations with programming

Discovery-based learning theory states that when constructing knowledge in a discovery-based environment, the learner will most likely struggle (Bruner, 1961).

Participants' frustrations from programming stemmed mostly from not understanding modular programming. This understanding was eventually engendered but only through much trial and error programming. This trial and error is known as debugging in computer programming. Thus, in addition to trying to manipulate an awkward remote controller to program the robot, students did not understand the subject matter, computer programming.

Weinstein suggested use of organizing as a cognitive strategy when learning cognitive subject matter, such as computer programming. As Anne implied, if group

members had been keeping track of trials and errors they may have been able to move forward with less trial and error and subsequently less frustration.

Also, with this lack of apriori knowledge, it was expected that groups would seek additional knowledge. In attempting to understand the lab objective, participants sought help through discourse with group members, viewing videos of robot capabilities online and engaging with the robot and its controller.

They could also engage in discourse outside the lab via a discussion board on the course' website, however, no participants took advantage of this offering. Additionally, they could also seek help from the lab assistant, however, only as a last resort in case they could not move forward. Out of all such knowledge, seeking avenues a member from each group specifically expressed their appreciation for having access to a hard copy of the owner's manual.

However, in seeking knowledge, participants found inappropriate material, such as a video showing how to program the robot to complete advanced tasks. Since there was an assumption among some group members that their group mates had certain knowledge and skills, attempts at completing advanced programming ensued.

Thus this study recommends curtailing advanced knowledge seeking activities, initially. Additionally, offer ways for teammates to ascertain their group mates skills. Such skills should be ascertained and confirmed before lab initiation to engender a better understanding of group mates' programming capabilities. The Computer Science Teachers Association (CSTA) has a set of standards relating to computer science

education (CSTA, 2011). An adaptation of these standards were devised to help facilitate minimizing assumptions about group mate experience.

The following recommendations are suggested to help minimize frustrations with programming:

1. Initially, novices should not divert from the stated lab problem. It should be mandatory to chose one of the carefully thought out lab options proposed by the instructor.
2. During pre-lab initiation activities, engender an appreciation for debugging, a necessary component in computer programming.
    a. During the peanut-butter-cracker logic activity before lab initiation, go through several trials of making a PBJ. During such trials, explain to students that in order to be successful in programming you will most likely have to go through many trials and errors before success.
    b. However, to minimize such trials, have students keep a day log capturing ideas, trials, successes, and failures. This sheet would consist of a four-column table capturing: ideas, trials, errors, and successes.
    c. Have one group member volunteer to undertake this task.
3. Determine group mates' experience by having group members answer and share with group mates the following questions in free-form format:

1. How would you define and evaluate algorithms by their efficiency, correctness, and clarity.
2. Explain the software life cycle as discussed in class lecture.
3. Classify programming languages based on their level and application domain as discussed in class lecture.
4. Explain the notion of intelligent behavior through computer modeling and robotics.

4. Viewing of videos should be in support of the stated lab objective. While students may look at videos displaying robotic capabilities, they should be provided with particular videos that might help aid understanding of their lab task. Thus while not limiting their viewing capabilities, educators should recommend applicable videos.
5. Provide a hard copy of instructional materials so that all group members can review and discuss at once.

Forestalling Premature Celebration

Because groups did not have an adequate understanding of what success looked like, both thought they had completed the lab objective on the first day, when in fact they had not. They lacked the ability to think about their thinking due to lack of knowledge. It appears that both groups sought outside affirmation because they did not understand what it meant to be successful. The assistant quickly told both groups that they had not succeeded. He also did not tell them how to complete the project, successfully.

Weinstein stated that metacognition and modifying can be used as a cognitive strategy. Metacognition is thinking about thinking. Thus, similarly to an assessment to determine if students have reviewed lab handouts to understand the technology, students should be assessed to determine if they understand the lab objective. The results of this study indicate the following recommendations:

1. Students need to be sure of the goal. Although stated in the lab handout, a pre-assessment should be given to ascertain if students understand the goal.
2. Group members need to think about their thinking and as the discovery-based theory states, students have to struggle to learn. However, to minimize struggling, assign a skeptic to serve as the group's thinker by questioning group's successes. This skeptic may also serve by using any number of participation indicator strategies to keep all members engaged, thus ensuring an authentic collaborative discovery-based learning environment. As discussed in chapter two, several groups did show total engagement.
3. In a discovery environment, do not rush in to offer advice once a group has failed. Allow students the time to think about how they might recover from such failure. On the other hand, do rush in to let groups know that they have not succeeded when they think they have.

Recommendations to Stay on Course

Both groups wanted to go beyond the stated lab objective at some point. Group A motivated by having the ability to be creative and go beyond from the onset, and Group B motivated after they had been successful in programming the robot. Hackman states that highly functioning group members would engage in task performance strategies that are not only appropriate to the work and the setting but also may go beyond. Additionally, Weinstein and Mayer states that elaboration facilitates understanding of cognitive skills. Going beyond a stated goal allows for elaboration, further understanding.

This study suggests the following to extend understanding:

1. Time should be allotted to allow students to explore and elaborate during the programming process, but stay on task of stated lab objective.
2. Time should be allotted to allow students to elaborate after the lab to extend their knowledge of computer programming.

Constructing Knowledge Collaboratively

Both groups worked well together throughout the two-week span. Seemingly, because members were allowed to work as they wished, either by proactively engaging in activities they wished to engage in or not engaging in activities that they were not comfortable with.

Nonetheless, some members wished other members would work differently. They wanted their group members to act in a different way, a way they thought would best make the group successful. As discussed in chapter four, this desire for different

work ethics may have impeded effective group work. In one case, the work of one group member propelled her group forward.

The results of this study indicate the following recommendations:

1. To facilitate groups working well together, allow group members to engage in the activity as they chose, initially. Working in such an unknown environment can be stressful and being allowed to work in those areas in which one is most comfortable might lesson anxiety. As time progresses and the unknown become more familiar, students may choose to engage in other activities. Do not assign hard role assignments initially. In a discovery environment, group members may not know what may peak their interest.
2. Instructors should get an idea of how students have dealt with adversity in the past by asking a pre-lab question: "Give an example of how you handled an adverse situation in the past". In analyzing student's response, determine student's self-efficacy by determining if the student thought they could have personally done something to affect the situation (Bandura, 1997). Although only one data point, it would give instructor some accounting of behaviors that might result in the group and inform on group placement.
3. Groups do not need a superstar; however, major insight can be gathered when any one member brings relevant knowledge from the past or by working alone. Encourage students to work outside their interactions in

the lab. Their use of personal metacognitive strategies may provide insight that not only informs them but also when shared, transfers knowledge to group mates

Implications for Research and Education

This study looked at how novice computer programming students engaged in a collaborative computer programming discovery-based learning environment. Findings may provide a paradigm for future research that wants to look at how tacit knowledge is transferred into explicit knowledge in a discovery-based learning environment. Most computer programming epistemologies privilege *individualized* tacit knowledge as a knowledge transfer mechanism for learning about computer programming. However, the impact of sharing *group* tacit knowledge should be given serious consideration as a knowledge transfer mechanism.

These findings could be made available for educators who might want to implement such an instructional strategy in their classroom, perhaps as a first course in computer programming. Understanding computer programming concepts in such a hands-on, discovery-based learning environment can serve as a prerequisite for students' understanding of more abstract, complex computer programming concepts.

Finally, this small qualitative case study provided a rich description of how students engaged in a group discovery based environment. However, what is unknown is how much knowledge actually transferred? What influence did collaborating in a computer programming have on one's collaborative skills? Further research on such questions might provide further validation of the effectiveness of this type of pedagogical

environment. Findings from this study may provide a framework for future research that looks at associating the impact of this type of learning environment on actual learning outcomes.

Conclusion

This study provided a bridge between computer programming and discovery based learning. This bridge was formed with rigor by viewing and transcribing over 900 minutes of videotapes, analyzing over 100 pages of transcription documents from which 381 raw codes resulted and five major themes emerged. Even though minimal teacher intervention was provided, the findings showed that while frustrating at times, students enjoyed learning about computer programming in a group collaborative discovery based environment. This is important because satisfaction in learning subsequently affects learning outcome effectiveness (Lo, 2010).

Hopefully, a pedagogical approach like this will be used in future computer programming courses. However, bringing the results of any qualitative study to bear on education policy can seem convoluted (Denzin & Lincoln, 2003). Nonetheless, it appears that computer fluency and collaborative skills are two of the *enormously important skills* today and similar to 20th century societal needs as chronicled by Graham, the emerging global interconnected information society necessitates a citizenry with such skills.

REFERENCES

Alessi, S. M., & Trollip, S. R. (2001). *Multimedia for learning methods and development.* Needham Heights, MA: Pearson Education.

Arango, F., Altuger, G., Aziz, E.-S., Chassapis, C., & Esche, S. (2008). Piloting a game-based virtual learning environment. *Computers in Education Journal* , 82-91.

Araujo, L., Filho, G., & Losada, M. (2005). Evaluating virtual learning communities using a nonlinear model. In V. Uskov (Ed.), *Proceedings of Computers and Advanced Technology in Education, Brazil, 470.* Oranjestad, Aruba: ACTA Press.

Attarzadeh, F. (2006). Empowering Students to become highly skilled professionals for the 21st Century Industries. *IJME-INTERTECH International Conference.* IEEE.

Baheti, P., Williams, L., Gehringer, E., & Stotts, D. (2002). Exploring pair programming in distributed object-oriented team projects. *Proceedings OOPSLA Educator's Symposium.* Seattle, WA.

Bandura, A. (2000). Exercise of human agency through collective efficacy. *Current Directions in Psychological Science , 9* (2), 75-78.

Bandura, A. (1997). *Self-efficacy: The exercise of control.* New York: W.H. Freeman.

Barkley, E. F., Cross, K. P., & Major, C. H. (2005). *Collaborative learning techniques.* San Francisco, CA: John Wiley & Sons, Inc.

Bauman, M.G. (2006). *Lessons from lost souls.* The Chronicle Review.

Bell, W. (2003). *Foundation of future studies: History, purposes, and knowledge* (Vol. 1). New Brunswick, NJ: Transaction Publishers.

BEST. (2010). *What is best?* Retrieved April 7, 2011, from BEST: http://best.eng.auburn.edu/b_about_best.php

Bevan, J., Werner, L., & McDowell, C. (2003). Guidelines for the user of pair programming in a freshman programming class. *Presented at Conference on Sofware Engineering Education and Training, 2002.*

Black, M. (2009). A processor design project for a first course in computer organization. *Computers in Education Journal , 20* (1), 95-103.

BLS. (2009). *EMPLOYMENT PROJECTIONS – 2008-18.* Retrieved March 16, 2011, from EMPLOYMENT PROJECTIONS – 2008-18: http://www.bls.gov/news.release/pdf/ecopro.pdf

Bonk, C. J., Lee, M. M., Kim, N., & Lin, M.-F. G. (2009). The tensions of transformation in three cross-institutional wikibook projects. *Internet and Higher Education , 12*, 126-135.

Bothamley, J. (2002). *Dictionary of theories.* Canton, MI: Visible Ink Press.

Bratzel, B. (2005). *Physics by design.* Knoxville, TN: College House Enterprises.

Bruner, J. (1961). *The act of discovery.* Harvard: Harvard Educational Review.

Bruning, R. H., Schraw, G. J., Norby, M. M., & Ronning, R. R. (2004). *Cognitive psychology and instruction.* Upper Saddle River, NJ: Prentice Hall.

Buckley, M., Kershner, H., Schindler, K., Alphonce, C., Braswell, J. (2004). Benefits of using socially-relevant projects in computer science and engineering education. *SIGCSE '04 Proceedings of the 35th SICCSE Technical Symposium on Computer Science Education*. Association for Computing (ACM).

Burstein, L., Freeman, H. E., & Rossi, P. H. (1985). *Collecting evalutaion data.* Beverly Hills, CA: SAGE Publications.

Bybee, R. W., Taylor, J. A., Gardner, A., Van Scotter, P., Powell, J. C., w, et al. (2006). *The BSCS 5E instructional model: Origins and effectiveness.* National Institutes of Health, Office of Science Education. Colorado Springs: BSCS.

Carspecken, P. F. (1996). *Critical ethnography in educational research.* New York, NY: Routledge.

Clark, R. E. (1994). Media will never influence learning. *Educational Technology Research & Development , 42* (2), 21-29.

CMU.EDU. (2007). *Carnegie Mellon University*. Retrieved April 7, 2011, from Alice Educational Software: http://www.cmu.edu/corporate/news/2007/features/alice.shtml

Combs, S. (2010). *Education and the Texas workforce.* Retrieved 2010 27-June from Window on State Government: http://www.window.state.tx.us/specialrpt/workforce/education.php

Corbin, J., Strauss, A. (2008). Basics of qualitative research (3e). Thousand Oaks: Sage Publications, Inc.

CSTA. (2011). *Computer Science Teachers Association*. Retrieved April 20, 2011, from New CS Curriculum Standards (Draft for Public Comment): http://csta.acm.org/includes/Other/CSTAStandardsReview2011.pdf

Dale, N., & Lewis, J. (2007). *Computer Science Illuminated* (Third ed.). Sudbury, MA: Jones and Bartlett Publishers.

D'Arcy, C. J., Eastburn, D. M., & Bruce, B. C. (2009). How media ecologies can address diverse student needs. *College Teaching* , 56-63.

Denner, J., Werner, L., Bean, S., & Campe, S. (2005). The Girls Creating Games Program: Strategies for engaging middle-school girls in information technology. *Frontiers: A Journal of Women Studies , 26* (1), 90-98.

Denzin, N. K., & Lincoln, Y. S. (2003). *Collecting and interpreting qualitative materials* (2nd ed.). Thousand Oaks, CA: Sage Publications, Inc.

Denzin, N. K., & Lincoln, Y. S. (1998). *The landscape of qualitative research: Theories and issues.* Thousan Oaks, CA: SAGE Publications, Inc.

Diamond, J. (2005). *Collapse: How societies choose to fail or succeed.* New York: Penguin Books.

Dick, W., Carey, L., & Carey, J. O. (2005). *The systematic design of instruction* (6th ed.). Boston, MA: Allyn and Bacon.

DOC. (2011). *Department of Commerce.* Retrieved May 17, 2011, from Digital literacy initiative fact sheet: http://www.digitalliteracy.gov/sites/digitalliteracy.gov/files/Digital_Literacy_Fact_Sheet_051311.pdf

Dougherty, K.J. (2001). *The contradictory college: The conflicting origins, impacts, and failures of the community college*. Albany, NY: State University of New York Press

D'Souza, C., Kazlauskas, A., & Thomas, T. (2009). Scaffolding strategies for teaching introductory programming. *Doctoral Student Consortium of Proceedings of the 17th International Conference on Computers in Education* (pp. 32-41). Hong Kong: Asia-Pacific Society for Computers in Education.

Duffy, T. M., & Cunningham, D. J. (1996). *Constructivism: Implications for the design and delivery of instruction. In D. Jonassen & S. Land (Eds.), Theoretical Foundations of Learning Environments (p. 32)* . New York: Macmillian.

ESA. (2011). *Entertainment software association*. Retrieved March 20, 2011, from Industry Facts: http://www.theesa.com/facts/index.asp

Evans, A., Martin, K., & Poatsy, M. A. (2009). *Technology in action* (5th ed.). Upper Saddle River, NJ: Pearson Education, Inc.

Fagerberg, J., Nelson, R., & Mowery, D. C. (2005). *The Oxford handbook of innovation.* Oxford University Press.

FIRST. (2011). *FIRST history*. Retrieved April 7, 2011, from FIRST: http://usfirst.org/aboutus/content.aspx?id=880&terms=250%2c000

FLL. (2011). *FIRST LEGO League*. Retrieved April 7, 2011, from Welcome to FLL: http://usfirst.org/Default.aspx

Forte, A., & Guzdial, M. (2005). Motivation and nonmajors in computer science:Identifying discrete audiences for introductory course. *IEEE Transactions on Education , 48* (2), 248-253.

Friedman, T. L. (2005). *The world is flat.* New York: Farrar, Strauss and Giroux.

Gagne, R. M. (1985). *The conditions of learning* (4th ed. ed.). New York: Holt, Rinehar & Winston.

Gale, K., Wheeler, S., & Kelly, P. (2007). Professional Identity and practice style in an online problem-based learning environment. *Quarterly Review of Distance Education , 8* (4), 297-307.

Gardner, H. (1999). *Intelligence reframed: Multiple intelligences for the 21st century.* New York: Basic Books.

Gates, E. A. (1999). *The application of language acquisition theory to programming concept instruction: Chunks versus programs from scratch.* Hattiesburg: USM.

Gilbert, J.K., Bulte, A.M., Pilot, A. (2011). Concept development and transfer in context-based science education. *International Journal of Science Education , 33* (6), 817-837.

Ginsburg, H., & Opper, S. (1969). *Piaget's theory of intetellectual development: An introduction.* Englewood Cliffs, NJ: Prentice-Hall, Inc.

Gokhale, A. A. (1995). Collaborative learning enhances critical thinking. *Journal of Technology Education , 7* (1), 22-30.

Gordon, D. (2011). Return to sender. *The journal: Transforming education through technology , 38* (3), pp. 31-35.

Graham, P. A. (2005). *Schooling America: How the public schools meet the nation's changing needs.* New York, NY: Oxford University Press.

Gura, M., & Kathleen, K. P. (2007). *Classroom robotics: Case studies of 21st century instruction for millenial students.* United States: Information Age Publishing.

Guzdial, M. (2003). A media computation course for non-majors. *ACM SIGCSE Bulletin.*

Hackman, J. (1990). *Groups that work (and those that don't).* San Francisco: Jossey-Bass.

Harris, J. B., & Hofer, M. J. (2011). Technological pedagogical content knowledge (TPACK) in action: A descriptive study of secondary teachers' curriculum-based, technology-related instructional planning. *JRTE , 43* (3), 211-229.

Harvard. (2010). *Harvard:Graduate School of Education*. Retrieved March 16, 2011, from Patricia Albjerg Graham: http://www.gse.harvard.edu/faculty_research/profiles/profile.shtml?vperson_id=441

Harvard. (1999). *J. Richard Hackman*. Retrieved March 19, 2011, from J. Richard Hackman: http://www.people.fas.harvard.edu/~hackman/

Harvey, B., & Wright, M. (1999). *Simply Scheme:Introducing Computer Science* (second ed.). Cambridge, Massachusetts: The MIT Press.

Hoganson, K. (2008). *Concepts in computing.* Sudbury, MA: Jones and Bartlett Publishers.

Howland, J., & Jonassen, D. H. (2003). *Learning to solve problems with technology: A constructivist perspective.* Upper Saddle River, NJ: Pearson Education, Inc.

James, S. (2005). *The wisdom of crowds.* New York: Random House.

Jonassen, D. H., & Land, S. M. (2000). *Theoretical foundations of learning environments.* Mahwah, NJ: Lawrence Erlbaum Associates.

Kagan, N., Krathwohl, D., & Miller, R. (1963). Stimulated recall in therapy using video tape: A case study. *Journal of Counseling Psychology , 10* (3), 237-243.

Kandi, T., Hirano, T., Eaton, D., & Ishiguro, H. (2004). Ineractive robots as social partners and peer tutors for children. *Human-Computer Interaction , 19* (1).

Kane, T. J., & Rouse, C. E. (1999). The community college: Educating students at the margin between college and work. *The Journal of Economic Perspectives , 13* (1), 63-84.

Keller, D. (2008). *Multiple literacies*. Retrieved 2008, from Multiple literacies: http://www.21stcenturyschools.com/Literacies/Multiple_Literacies.htm

KISS. (2009). *The Botball difference*. Retrieved April 7, 2011, from Botball: http://www.botball.org/botball_difference

Kolb, L. (2008). *Toys to tools:Connecting student cell phones to education.* Washington, D.C.: International Society for Technology in Education (ISTE).

Kozma, R. B. (1994). Will media influence learning? Reframing the debate. *Educational Technology Research & Development. , 42* (2), 7-19.

Kubler, B., & Forbes, P. (2005). *Student employability profiles.* UK: The Council for Industry and Higher Education.

Lego. (2010). *Lego mindstorms*. Retrieved 10 20, 2010, from Lego mindstorms: http://mindstorms.lego.com/en-us/whatisnxt/default.aspx

Leonard, D. C. (2002). *Learning theories: A to Z.* Westport, CT: Greenwood Press.

Linn, M. C., Sloane, K. D., & Clancy, M. J. (2006). Ideal and actual outcomes from precollege pascal instruction. *Journal of Research in Science Teaching , 24* (5).

Liveops. (2011). *Liveops*. Retrieved May 16, 2011, from Overview.

Lo, C. C. (2010). How student satisfaction factors affect perceived learning. *Journal of the Scholarship of Teaching and Learning , 10* (1), 47-54.

Louis, A. (2010). *Does discovery-based instruction enhance learning?* Dissertation, City University of New York.

Matzko, S., & Davis, T. (2006). Pair design in undergraduate labs. *Consortium for Computing Sciences in Colleges* , 123.

Mayer, R. E. (2001). *Multimedia learning.* Cambridge, MA: Cambridge University Press.

Merriam (2001). *Qualitative research and case study applications in education: Revised and expanded from Case Study Research in Education.* San Francisco: Jossey-Bass

Michaelsen, L. K., Knight, A. B., & Fink, L. D. (2002). *Team-based learning: A transformative use of small groups in college teaching.* Westport: Greenwood Publishing Group, Inc.

Microsoft. (2002). *Microsoft computer dictionary.* Redmond, WA: Microsoft Press.

Mishra, P., & Koehler, M. J. (2006). Technological pedagogical content knowledge: A framework for teacher knowledge. *Teachers College Record , 108* (6), 1017-1054.

Moreno, R. (2009). Cognitive load theory: More food for thought. *Instructional Science , 38* (2), 135-141.

NCSU. (2010). *Pair learning: The use of pair programming in education*. Retrieved March 18, 2011, from Pair learning: The use of pair programming in education: Pair Learning: The Use of Pair Programming in Education

Nelson, L. (1999). *Collaborative Problem Solving.* Mahwah, NJ: Lawrence Erlbaum Associates.

NLN. (2010). *Maria's NLN page*. Retrieved April 14, 2011, from myNational Lab Network: http://my.nationallabnetwork.org/discoveredlearning

NSF. (n.d.). Retrieved 2011, from http://www.technologyreview.com/computing/32429/?a=f

NSF. (2008). *NSF and the birth of the internet National Science Foundation*. Retrieved March 17, 2008, from National Science Foundation: Where discoveries begin: http://www.nsf.gov/news/special_reports/nsf-net/home.jsp

ODesk. (2011). *Changing how the world works*. Retrieved May 16, 2011, from Changing how the world works: https://www.odesk.com/w/odesk_story

Papert, S. (2005). Teaching children teaching. *Contemporary Issues in Technology and Teacher Education , 5* (3/4).

Pastel, R. (2006). Michigan Technological University. Consortium for Computing Sciences in Colleges.

Pastel, R. (2006). Student assessment of group laboratories in a data structures course. *Consortium for Computing Sciences in Colleges* , 221.

Pederson, J. E., & Digby, A. D. (1995). *Secondary schools and cooperative learning.* New York: Garland Publishing, Inc.

Pierson, M. E. (1999). *Technology integration practice as a function of pedagogical expertise.* Arizona State University.

Plomp, T., & Ely, D. P. (1996). *International encyclopedia of educational technology* (2nd ed.). Cambridge, UK: Cambridge University Press .

Postner, L. (2003). *What's so hard about learning to program? A cognitive and ethnographic analysis of beginning programming students.* Dissertation.

Preston, D. (2005). Pair programming as a model of collaborative learning: A review of the research. *Consortium for Computing Sciences in Colleges* , 39.

Raby, R. L., & Tarrow, N. (1996). *Dimensions of the community college: International, intercultural, and muticultural perspectives.* New York: Garland Publishing, Inc.

Ramadhan, H. A. (2000). Programming by discovery. *Journal of Computer Assisted Learning , 16,* 83-93.

ROBOFEST. (2011). *About ROBOFEST and FAQ.* Retrieved April 7, 2011, from ROBOFEST: http://robotfest.net/

Rodrigo, M. T., & Baker, R. S. (2009). Coarse-Grained detection of student frustration in an introductory programming course. *International Conference on Education Research (ICER)* (pp. 75-79). New York: ACM.

Rudestam, K. E., & Newton, R. R. (2007). *Surviving your dissertation: A comprehensive guide to content and process.* Thousand Oaks, CA: Sage Publications, Inc.

Saunders, D. (2009). Podcasting in the classroom using thought provoking dialogue. Retrieved April 20, 2009 from youtube.com/watch?v=wL1bX1gepEc.

Schunk, D. (2004). *Learning theories, an educational perspective.* Upper Saddle River, New Jersey: Pearson Prentice Hall.

Shank, R., Berman, T., & Macperson, K. (1999). *Learning by doing.* Mahwah, NJ: Lawrence Erlbaum Associates.

Shelly, G., Cashman, T., Gunter, R., & Gunter, G. (2004). *Teachers discovering computers. Integrating technology in the classroom* (3rd ed.). Boston, MA: Thomson Course Technology.

Sherrell, L. B., & Robertson, J. J. (2006). Pair Programming and Agile software development: Experiences in a college setting. *Consortium for Computing Sciences in Colleges* , 145.

Smith, P. L., & Ragan, T. J. (1999). *Instructional design* (2nd ed.). Hoboken: John Wiley & Sons, Inc.

Smith, S. M., & Woody, P. C. (2000). Interactive Effect of Multimedia Instruction and Learning Styles. *Teaching of Psychology , 27* (3), 220-223.

Squire, K. D., DeVane, B., & Durga, S. (2008). Designing centers of expertise for academic learning through video games. *Theory Into Practice , 47*, 240-241.

Squire, K. (2007). Open-Ended video games: A model for developing learning for the interactive age. *DMAL , 14* (5).

Stake, R. (1995). *The art of case research.* Thousand Oaks, CA: Sage Publications.

Swan, K., Hofer, M., & Swan, G. (2011). Examining authentic intellectual work with a historical digital documentary inquiry project in a mandated state testing environment. *Journal of Digital Learning in Teacher Education , 27* (3).

Sykes, E.R. (2007). Determining the effectiveness of the 3D Alice programming environment at the Computer Science I level. *Journal of Educational Computing Research.*

Tannehill, D. B. (2009). *Andragogy: How do post-secondary institutions educate and service adult learners?* University of Pittsburgh.

Tannenbaum, P. H. (1959). The birth of FIRO. *PsycCRITIQUES , 4* (10), 312-314.

THOMAS. (2011). *U.S. Government Printing Office.* Retrieved May 16, 2011, from One Hundred Ninth Congress: http://frwebgate.access.gpo.gov/cgi-bin/getdoc.cgi?dbname=109_cong_bills&docid=f:s250enr.txt.pdf

Thurlow, C., Lengel, L., & Tomic, A. (2004). *Computer mediated communication.* Thousand Oaks, CA: SAGE Publications USA.

Treagust, D., Chittleborough, G., Mamiala, T. (2002). Students' understanding of the role of scientific models in learning science. *International Journal of Science Education, 24 (4),* 357-368

Tutty, J., Sheard, J., & Avram, C. (2008). Teaching in the current higher education environment:Perceptions of IT academics. *Computer Science Education , 18* (3), 171-185.

Unger, H. G. (2007). *Encyclopedia of American education F-P* (Third ed.). New York: Facts on File.

VanDeGrift, T. (2004). Coupling pair programming and writing: Learning about students' perceptions and processes. *SIGCSE* , 2-6.

VEX, R. (2010). *VEX Robotics Design System*. Retrieved April 7, 2011, from VEX Education Overview: http://www.vexrobotics.com/education/

Wang, E. (2004). *Engineering with LEGO bricks and RoboLab.* Knoxville, TN: College House Enterprises.

Warner, A. R. (2008). Seminar in curriculum and instruction. *CUIN 8360* . Houston, TX: University of Houston.

Waters, J. K. (2011). Rock 'em, sock 'em! *THE* , 30-35.

Webopedia. (2011). *Moore's Law*. Retrieved April 4, 2011, from Webopedia : http://www.webopedia.com/TERM/M/Moores_Law.html

Weinberger, A., Ertl, B., & Fischer, F. (2005). Epistemic and social scripts in computer-supported collaborative learning. *Instructional Science , 33* (1), 1-30.

Weinstein, C. F., & Mayer, R. F. (1986). *The teaching of learning strategies: In M.C. Wittrock (Ed.), Handbook of research on teaching (3rd ed.) (pp. 315-329).* New York: Macmillan.

Wikipedia. (2010). *Wikipedia*. Retrieved March 24, 2011, from Wikipedia: http://en.wikipedia.org/wiki/Gamer

Williams, L. A. (2000). *The collaborative software process.* University of Utah. University of Utah.

Williams, L., & Upchurch, R. L. (n.d). *In support of student-pair programming.* North Carolina State University.

Wowwee. (2011). *wowwee.com*. Retrieved september 16, 2008, from wowwee.com: www.wowwee.com

Zacharis, N. Z. (2010). Measuring the effects of virtual pair programming in an introductory programming JAVA course. *Technological Educational Institute of Piraeus* .

Zhen-Jia You, C.-Y. S.-W.-J.-D. (2006). A Robot as a Teaching Assistant in an English Class. *Sixth IEEE International Conference on Advanced Learning Technologies (ICALT'06)* , 87-91.

APPENDIX A COURSE SEMESTER SCHEDULE

Table 3.

Course Semester Schedule

Week	Chapters	Concepts	Skills
1	Course Introduction, Chapter 1	Overview	Navigating myitlab
2	Chapter 11	Databases	ACCESS
3	Chapter 2	Parts of a computer	ACCESS
4	Chapter 3	Using the Internet	ACCESS
5	Chapter 4	Application Software	ACCESS Presentation &Exam
6	Chapters 5&6	System Software; Hardware	EXCEL
7	Chapter 7	Networking and Security	EXCEL
8	Chapter 8	Mobile Computing	POWERPOINT
9	Chapter 9	System Hardware	POWERPOINT
10	Course Project	Computer Programming	POWERPOINT/EXCEL
11	Course Project	Computer Programming	Combo Excel/PP Presentation
12	Course Project	Computer Programming	WORD
13	Chapter 12	BTS: Internet	WORD
14	Review Semester Material	Review for Final Exam	Course Project Presentation
17	FINAL EXAM		Comprehensive (Lecture)

APPENDIX B ASSISTANT'S CONTRACT

APPENDIX B Assistant's Contract

To: Phillip Price
From: Maria T Earle
Sub: Research Assistant Responsibilities
Date: 10/28/2010

Dear Mr. Price:

Thank you so much for considering to work as my assistant for my upcoming research study at Lone Star College - Montgomery. The purpose of this letter is to confirm your availability to work as my research assistant, provide details of your responsibilities, and provide compensation details.

First, I am doing research with students in classes I am teaching at Lone Star – Montgomery this fall. The dates of the study are Mondays and Wednesdays, November 8-17, 2010; and the two classes meet: 8:00am – 10:20 am; 10:30am – 12:50pm. The basic objective of the research is to see how students might understand computer programming concepts by interacting with fellow peers and a robot, the same robot you used when you completed your lab in my class. The research will be done in a conference room (hereafter known as the lab) in building MC-F, First floor, Rm. #141. This room is in the Library at Lone Star - Montgomery College.

Basically, your main responsibilities will be taking place on four days, November 8,10,15,18 from 7:30am – 1:00 pm. In addition, you will be required to meet with the researcher twice prior to the beginning of the project, once on Thursday, October 28 and on Friday, November 5th and once after the completion of the project on Thursday, November 18. Given those four days and the pre/post research meetings required with the researcher, you will be compensated roughly $100/day for a total of $500.

Below is a detailed listing of the dates and times you would have to be available for the research study:

Thursday, Oct. 28th	Meet with Ms Earle, the researcher at Lone Star - Montgomery
Wednesday, Nov. 3rd, ; 10am	Collect participant consent forms in the classroom
Friday, Nov. 5th	Meet with Ms Earle to go over final preparations for research
Monday, Nov. 8th	First research day; Meet first group of students in classroom, then to lab
Wednesday, Nov. 10th	Second research day; Meet first group of students in classroom, then to lab
Monday, Nov. 15th	Third research day; Meet first group of students in classroom, then to lab
Wednesday, Nov. 17th	Fourth research day; Meet first group of students in

classroom, then to lab
Thursday, Nov. 18 Post research meeting with Ms. Earle at Lone Star – Montgomery.

Prior to the beginning of the research study, on Monday, Nov.1st, the researcher, Ms. Earle, will go over the goals of the research with her students during class time. She will also hand out consent forms. Students will be told to bring consent forms back to class on Wednesday with strict instruction that only the research assistant, you, can collect the forms.

On Wednesday, Nov 3rd. during the last 10 minutes of class1 and the first 10 minutes of class2, the research assistant will collect the consent forms. The instructor will not be in the classroom at this time. After consent forms have been collected, the assistant will have to place students in groups. Although the lab project is a mandatory component of the course, students can opt-out of having their lab work videotaped and counted as research data. Those students who have opted out of having their project data entered as part of the research data will be grouped into one separate group.

There should be 4 groups and they should be labeled A, B,C, D. There are ~ 20 students in the class. They should be broken up into 4 groups of 5 students each. One of the 4 groups should be reserved for abstainers, those who opt out of having their information entered in research and/or those who have had previous experience with the robot.

During the actual research, each group (A, B, C, D) will rotate in the lab for ½ hour each with 5 minutes of travel time. During this 5 minute reprieve, the assistant will prepare the lab for the next group. These would be group departure times for the 8am class:

Group A: 8:00
Group B: 8:35
Group C: 9:10
Group D: 9:45; they should take their things with them, they will not be coming back to the classroom.

These are the group departure times for 10:30am class:

Group A: 10:30am
Group B: 11:05am
Group C: 11:40am
Group D: 12:15pm; they should take their things with them, they will not be coming back to the classroom.

The assistant will meet the first group for each class in the classroom and then proceed to the lab. All subsequent groups will go to the lab once the previous lab group returns to class.

In the lab, a video camera will be available for videotaping the lab activity. In addition, a laptop computer cart will be available for students to go to the manufacturer's site to review information about the robot. While students are in the lab, the video tape will need to be turned on for consenting groups. Before the abstainer group enters the lab, the video camera will have to be turned off. During the videotaping, the assistant will need to note the times in a research journal that student group work begins and ends.

The assistant's responsibility while the students are in the lab is one of facilitator. You will not have to teach participants. Although, if students are stuck, you will be allowed to provide minor suggestions ito keep the research project moving forward. On the last day of the lab project, students will demonstrate their lab to you. In addition, participants will complete a score sheet for each of their group mates. This score sheet will be placed in a sealed envelope. The researcher will not view this sheet or the videos until time the end of the semester.

If you agree to the requirements of this letter, and working as Ms. Earle's research assistant, please print and sign your name below.

Once again, I really appreciate you taking the time to work with me on this research. I am sure it will prove invaluable as I go forward towards completing my doctoral research.

Print Name _____

Signature_____

Date_____

Phillip Price Contact Info: C – (832)764-6387; phillip1989@sbcglobal.net

Sincerely,

Maria Earle
H: (281) 419-2560
C: (408) 394-6520
marie.t.earle@lonestar.edu

APPENDIX C PEER DEBRIEFER CONTRACT

APPENDIX C Peer Debriefer Contract

To: Dr. Julie Trenor
Fm: Maria T. Earle
Sub: Earle Research - Peer Debriefer/Coder
Date: 2/4/2011

Dear Dr. Trenor:

Thank you for taking the time to help me with my research. I sincerely appreciate your time. This letter will provide a brief overview of my research, discussion of duties, and finally, payment arrangements.

My research is looking at the experiences of students learning about computer programming with the aid of a robot. Last fall, students from two sections of my classes at Lone Star college were solicited to participate in the first phase of my research, the observation phase. Eight groups of three students each had to program a robot to solve a program. These students are non-computer science majors and had never taken a formal computer programming course. The groups were videotaped while they worked.

These observation videotapes were then transcribed. From these observation transcriptions, potential interviewees were identified for the next phase of the study, the interview phase. Of the phase one participants, five agreed to be interviewed. I developed a unique interview protocol for each of the five interviewees. The one hour interviews were audio taped and are now being transcribed.

The next phase involves coding and determining emerging themes; this is where your expertise comes in. Your duties will be as follows for the five transcribed documents:

1. develop low-level codes (Carspecken, Ch.9)
2. develop emerging themes
3. report to PI on your findings as needed

Basically, your main responsibilities will consist of analyzing and coding the five transcribed interview documents. In the meantime, I am also analyzing and coding the transcriptions and will later compare my codes with yours. While I am using Phil Carspecken's coding methodology (chapter 9) to analyze and code, you can use whatever coding methods you like. You will also have each interviewees unique interview protocol to help guide you during the analysis and coding process. You will need to meet with the PI, Maria Earle, once before transcribing begin and once after.

You should expect to see the first transcription by Monday, February 6th via email or fax, whatever you prefer. And then you will receive another transcription approximately every two days. Thus, you will have all five transcriptions over a ten day period. The last transcription would then arrive Feb.16th. All coding needs to be completed and returned to the PI, Maria Earle, by Monday, February, 21st. As you complete analyzing one document, please send to the PI.

You will be compensated $100/document, for a total of $500. A bank check will be mailed to you once work, as discussed above, is complete. If you agree to the stipulations of this letter, please print and sign your name below and then fax. If you prefer, you can mail it.

Once again, I really appreciate you taking the time to work with me on my research. I am sure your engineering and qualitative research experience will prove invaluable to my research as I go forward.

Name Julie Martin Trenor
Signature Julie Martin Trenor
Date 2/6/11

Sincerely,
Maria T. Earle
H: (281) 419-2560; C: (408) 394-6520
F: (440) 809-2880
marie.t.earle@lonestar.edu
mtearle@uh.edu

APPENDIX D WORKSHEET TEMPLATE

APPENDIX D Worksheet Template

Problem Statement (basic moves, vision, sound)

Algorithm & Flowchart

Script

Main program

SH1+SH2+c	enter main programming mode
Sh1+Sh2+square	clear contents of memory
Press STOP (square)	
SH1+SH2+x	Store main programming and execute

Vision Subprogram

Sh1+Sh2+b	enter vision programming mode
Sh1+Sh2+square	clear contents of memory
Press STOP (square)	
SH1+SH2+x	Store vision programming and execute

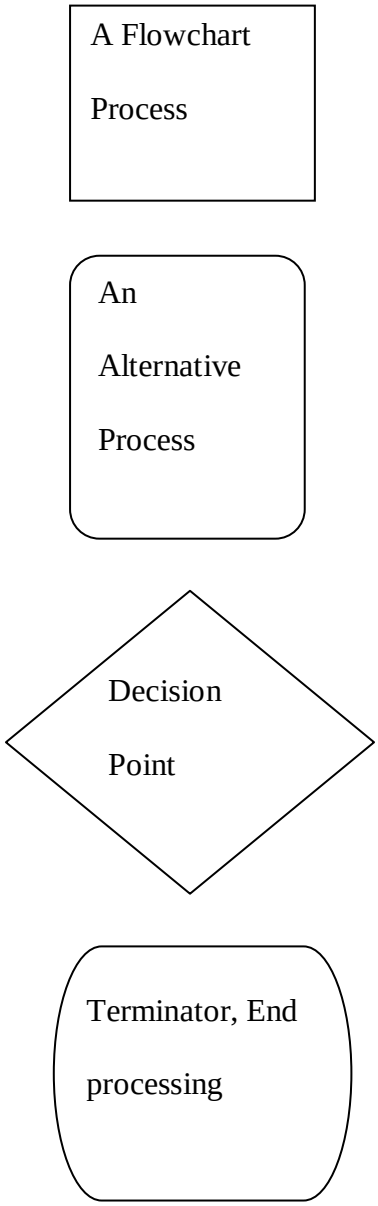
Sound Subprogram

Sh1+Sh2+a	enter sound programming
Sh1+Sh2+square	clear contents of memory
Press STOP (square)	
SH1+SH2+x	Store sound programming and execute

Run Instructions (Sh1+Sh2+x)
Face robot towards a white wall

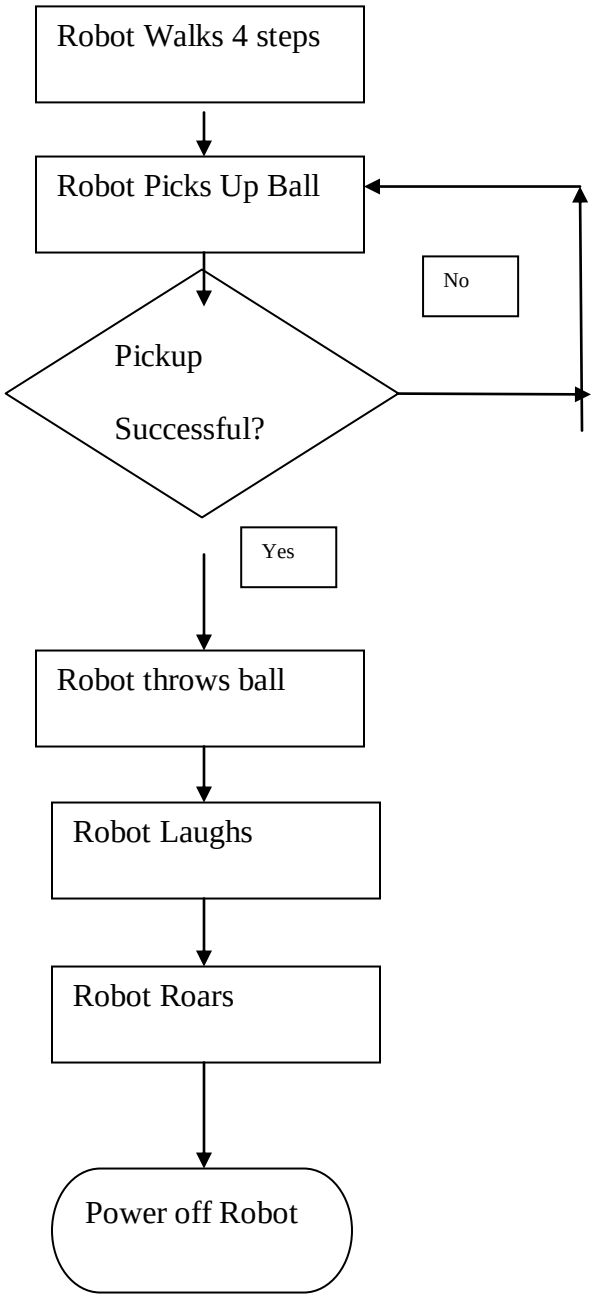
Creating Flowcharts

1. Open Microsoft Word
2. From menu, choose Insert
3. Choose Shapes
4. Choose a Flowchart symbol to represent steps in algorithm.

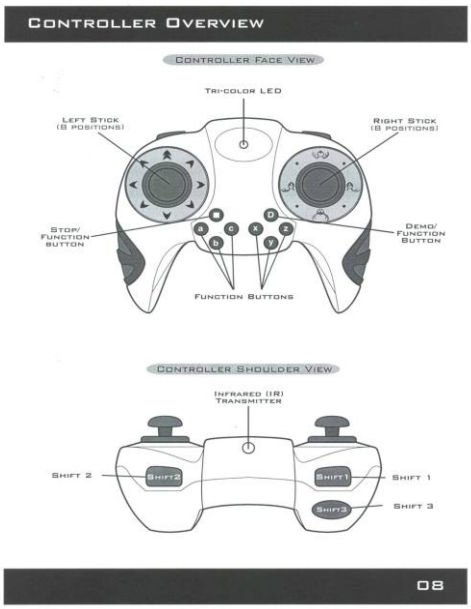


RoboSapien

Sample Flowchart



APPENDIX E RoboSapien v2.0


CONTROLLER OVERVIEW
CONTROLLER FACE VIEW
TRI-COLOR LED
LEFT STICK (8 POSITIONS)
RIGHT STICK (8 POSITIONS)
STOP/ FUNCTION BUTTON
DEMO/ FUNCTION BUTTON
FUNCTION BUTTONS
CONTROLLER SHOULDER VIEW
INFRARED (IR) TRANSMITTER
SHIFT 2
SHIFT 1
SHIFT 3
08

APPENDIX F STUDENT DELIVERABLES

APPENDIX G ASSISTANT'S FIELD NOTES

Table 4

Assistant's Field Notes

Table X

Assistant Field Notes: High level summary of group activity – 8 original groups

Participant Groups	Day One	Day Two	Day Three	Day Four	Programming Success
Group A	Realized that they have to put the subroutine in the main program	Realized that the subroutines have to be in main program	Has a good grasp on the programming and works well together	No entry	YES
Group B	No entry	Has an unorthodox way of using the vision system. Struggling with programming the robot	Argued a little bit, but understood what they wanted him to do.	No entry	YES
Group C	Not a lot of communication, but progressed slowly	Better communication and seemed to be progressing a lot better than the previous day	Having a hard time trying to figure out how to..utilize sound and vision	Ran the way they wanted but did not use the sound and vision program	NO
Group D	No entry	Figured out that you have to input the info in the subroutines	Had run instructions when they walked in but had some problems	No entry	NO
Group E	They are getting the hang of the vision on range of the robot	Having trouble inputting the information in program mode	Communicating well with each other	Everything worked well	YES
Group F	Asked a lot of questions, seemed confused on what to do.	Picked up on the programming quickly	They communicated well and discussed well. Was able to write out run instructions and run it correctly	No entry	YES
Group G	Figured out a lot of the moves. Demonstrated main programming	Switched from just throwing to having it bowl. Picked up the basics of the programming	Had trouble understanding the vision sub program and had a lot of questions about it	Did not follow run instructions perfectly, there was added instructions.	NO
Group H	Had trouble getting started, but slowly progressed	Having trouble getting into program mode	Started to understand the programming but not quite grasping the subprograms	No entry	NO

APPENDIX H INTERVIEW RAW CODES - TODD

Table 5. Todd's Low Level Themes

Low level code number	RESULTING LOW LEVEL THEMES	Raw Code Number Association
1	figure capabilities of robot	1,2,3,4,21-25,28-32, 34-37, 46,47,49-52,56,57,59,64
2	robot expectations	18-20,33,40,48,84
3	going beyond	6-10, 13-16, 26,27,38,39,67
4	team mate expectations	16,17
5	seeking knowledge	41
6	bringing meaning to programming with own terminology	43-45, 53
7	how to program	54,55,58,61,62,65,66,69-78,83
8	going back to previous step	60,68
9	perceptions on the lab	87-90
10	controller challenges	5
11	programming expectations based on prior experience	11,12
12	personal lab perception	91
13	thought done	
14	signs of adversity	92
15	The way I work	

APPENDIX I INTERVIEW RAW CODES – TANYA

Table 6. Tanya's Interview Raw Codes

Low level code number	RESULTING LOW LEVEL THEMES	Raw Code Number Association
1	figure capabilities of robot	2,3,7,8,9,11,15,17,18,19,20,22,24,26,27,28,29,50,64
2	robot expectations	23
3	going beyond	25,32,34,40,68,69,70,88,89,90,91,98
4	team mate expectations	71,75,84,85,86,94,95,96,114
5	seeking knowledge	51,56,65,66,67,72
6	bringing meaning to programming with own terminology	
7	how to program	1,4,5,6,41,52,57,63,73,74,76,80,81,82,87,100,101,102,103, 104,105,106,107
8	going back to previous step	10,13,16,21,38,78,83,92,99,113
9	perceptions on the lab	
10	controller	14

	challenges	
11	programming expections based on prior experience	
12	personal lab perception	
13	thought done	12
14	signs of adversity	59,60,61,62,77,79
15	The way I work	

APPENDIX J INTERVIEW RAW CODES - KATHY

Table 7. Kathy's Low Level Codes

Low level code number	RESULTING LOW LEVEL THEMES	Raw Code Number Association
1	figure capabilities of robot	65,78,92
2	robot expectations	5,6,8,28,33,42,44,53,54,66,68,77,80
3	going beyond	36
4	team mate expectations	2,7,10,11,12,13,14,16,18,19,20,21,22,23,32,43,44,45,46,,47,48,49,62,63,64,81,82,83,90,104,107,108,109,110,113,115,116,121,122,123,124,125,125,126,135,136,137,138
5	seeking knowledge	84100101111
6	bringing meaning to programming with own terminology	
7	how to program	9,40,41,56,57,60,67,70,71,98,99,105,106,127,128,129,130,131,132,133
8	going back to previous step	27,29,31,34,61,79,102,103,117,118,119,134

9	perceptions on the lab	3,139,140,141,142,143,144,145,146
10	controller challenges	4,85,86,87,89
11	programming expections based on prior experience	
12	personal lab perception	
13	thought done	
14	signs of adversity	91,93,94,96,97,114
15	The way I work	95

APPENDIX K INTERVIEW RAW CODES - ELAINE

Table 8. Elaine's Interview Raw Codes

Low level code number	RESULTING LOW LEVEL THEMES	Raw Code Number Association
1	figure capabilities of robot	2,3,7,8,9,11,15,17,18,19,20,22,24,26,27,28,29,50,64
2	robot expectations	23
3	going beyond	25,32,34,40,68,69,70,88,89,90,91,98
4	team mate expectations	71,75,84,85,86,94,95,96,114
5	seeking knowledge	51,56,65,66,67,72
6	bringing meaning to programming with own terminology	
7	how to program	1,4,5,6,41,52,57,63,73,74,76,80,81,82,87,100,101,102,103,104,105,106,107
8	going back to previous step	10,13,16,21,38,78,83,92,99,113
9	perceptions on the lab	
10	controller challenges	14

11	programming expections based on prior experience	
12	personal lab perception	
13	thought done	12
14	signs of adversity	59,60,61,62,77,79
15	The way I work	

APPENDIX L INTERVIEW RAW CODES - ANNE

Table 9. Anne's Interview Raw Codes

Low level code number	**RESULTING LOW LEVEL THEMES**	**Raw Code Number Association**
1	figure capabilities of robot	1,2,3,4,5,6,34,36,37,46
2	robot expectations	10,6,13,9
3	going beyond option	1,40
4	team mate expectations	10,61,63,64,66,67,70,71,72,73,75,86,87,89,109,110,111,124,125,126,129,130,134,135,136,137
5	**seeking knowledge**	**43,44,45,138**
6	bringing meaning to programming with own terminology	
7	how to program	7,8,9,11,12,13,14,15,28,29,30,31,32,35,39,40,41,42,47,49,50,51,52,53,54,55,56,57,58,59,60,76,78,82,83,84,91,92,93,94,95,96,97,98,99,100,101,102,103,104,107,108,112,115
8	going back to previous step	23,24,25,27,79,80,85,90,132
9	perceptions on the lab components	
10	controller challenges	

11	programming expectations based on prior experience	
12	personal lab perception	113
13	thought done	16,17,18,19,20,21,22,26,48,105
14	signs of adversity	
15	The way I work	127

APPENDIX M REFLECTION PAPER GUIDELINES

APPENDIX M Participant Reflection Paper Guidelines

WORD Independent Project (IP): Personal Reflection
COSC/ITSC 1401 - Earle 2010

Reflection Paper - Class Lab Project

Write a 2- to 4-page paper in which you reflect on your personal learning experience during the Lab. The paper should use the following format: Times New Roman, 12, double-spaced. Here are some questions you may want to consider:

1. What learning experiences did you find most useful during the Lab?
2. What were your feelings working with the robot?
3. What were your feelings working with your group-mates?
4. What are some of the possible implications of your work in the lab on other classes you take?
5. What elements of the Lab should be retained?
6. What improvements would you suggest?

Drop your final paper in the Independent Labs Drop box on myitlab.com.

Due Date – Wednesday, November 24, 2010

APPENDIX N THEMES HIERARCHY

Table 10. Emergent Themes Hierarchy

Emergent Theme Numbers	EMERGENT THEMES	Low Level theme number association	Low Level theme number association	Low Level theme number association
1	**frustrations with technology - the robot, its capabilities and its controller**	**1**	**2**	**10**
2	**Going Beyond**	**3**		
3	**team mate expectations**	**4**		
4	**seeking knowledge**	**5**		
5	**bringing mearning to programming**	**6**		
6	**frustrations with programming**	**7**	**8**	
7	**programming expectations**	**11**		
8	**thought done**			
9	**signs of adversity**			

APPENDIX O STUDENT DATA SHEET

APPENDIX O Student Data Sheet

Student Data Sheet

Last Name:

First Name:

Preferred Name:

Street Address:

City:

Zip:

Daytime Phone:

Evening Phone:

Preferred e-mail Address:

Social networking sites (i.e. MySpace):

Alternate means of contacting you:

1. Tell me about your semester.

A. How many hours are you taking?

B. What other courses are you taking?

C. Do you work? How many hours per week?

2. Are you new to Montgomery College?

3. Have you decided on a degree/certification program?

4. How much experience do you have with computers?

None □Some □Lots □

Please explain:

I have read the syllabus and agree to comply with its conditions.

I further certify that all work turned in will be my own, not copied, borrowed or done by another individual.

Signed: _____Date: _____

APPENDIX P DEMOGRAPHICS AND EXPERIENCE SHEET

Table 11.

Interviewees' Demographics and Computer Experience

	Ethnicity	AGE	Experience with computers	Experience with computer programming or robots
GROUP A				
Kathy	**African American**	**47-55**	**Some**	**None**
Tanya	**Caucasian**	**40-46**	**Lots**	**None**
Todd	**Caucasian**	**18-20**	**Lots**	**Some**
GROUP B				
Anne	**Hispanic**	**40-46**	**Some**	**None**
Elaine	**Caucasian**	**18-20**	**lots**	**None**

Interviewee Demographics and Experience Sheet

EARLE RESEARCH

"Feelings towards and experience with computer programming and robots".

In general, what were your initial feelings about this project, when you first heard about it in class.

Have you had any experience with computer programming before this class? If so, how?

What were your feelings towards computer programming before this project?

Have you had any experience with robots before this class? If so, how?

What were your feelings towards robots before this project?

Participant Background Sheet

(please circle or fill in the blank)

Name

_____

Age

18 – 20

21- 25

26-31

32-39

40-46

47-55

>55

Gender

Female or Male

Ethnicity

Classification

First in family to attend college - YES or NO

APPENDIX Q PEER DEBRIEFER'S MAJOR THEMES

APPENDIX Q Peer Debriefer's Themes

1. Participants discovered the capabilities and limitations of the robot and its programming through trial and error, which often was frustrating at the time

2. Trial and error led the participants to experience frustration during the project.

3. In hindsight (when viewing the video), participants were able to articulate their misconceptions and mistakes.

4. Specifically, participants described learning the following aspects of programming "the hard way" subroutines, ROM, main program and calling, and debugging

5. Once introduced to formal programming language with which participants had practical experience, they adopted it quickly and correctly used it during the interview.

6. Participants found the manuals useful when they utilized them; however some participants desired to experiment with robot first.

7. Multiple team dynamics were in play during the assignment.

8. Participants often personified the robot (e.g. referring to it as "he" and talking to it).

9. Teams or team members sometimes expressed wanting to go "above and beyond" the assignment and acquired the resources to do so, whether or not they were successful.

10. Participants compared themselves/their project to presumed "expert" programmers via YouTube videos; this sometimes led to grandiose ideas and/or feelings of frustration that they could not accomplish more advanced tasks.

11. Participants experienced a range of emotions throughout the assignment, frustration from the onset to a sense of satisfaction with their accomplishment: One participant in particular expressed feelings of accomplishment that he/she believed would carry over into other aspects of his/her life (e.g. after accomplishing an "impossible" task).

BLANK