Effectiveness of Using Digital Game Playing in a First-Level Programming Course

by
Sandra Westcott

Submitted in partial fulfillment
of the requirements for the degree of
Doctor of Professional Studies
in Computing

at

Ivan G. Seidenberg School of Computer Science and Information Systems
Pace University

December 2008

INFORMATION TO USERS

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleed-through, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

UMI®

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

We hereby certify that this dissertation, submitted by Sandra Westcott, satisfies the dissertation requirements for the degree of *Doctor of Professional Studies in Computing* and has been approved.

_____-_____

Dr. Joseph Bergin Date
Chairperson of Dissertation Committee

_____-_____

Dr. Fred Grossman Date
Dissertation Committee Member

_____-_____

Dr. Charles Tappert Date
Dissertation Committee Member

Ivan G. Seidenberg School of Computer Science and Information Systems
Pace University 2008

Abstract

Effectiveness of Using Digital Game Playing in a First-Level Programming Course

by
Sandra Westcott

Submitted in partial fulfillment
of the requirements for the degree of
Doctor of Professional Studies
in Computing

December 2008

In this high tech world of globalization it is paramount that students know how to think critically, and know how to identify, analyze, and solve problems quickly and effectively. Computer programming courses can provide students with a good foundation in these basic skill sets however, in many U. S. colleges and universities student enrollment in computer related majors is declining. In the fall of 2006 only 1.1 percent of incoming freshmen expressed any interest in computer science as a major. Enrollment in Computer Science needs to increase if we are to remain competitive.

Often students enrolled in introductory computer programming courses find the subject difficult. Several studies have concluded that even after students successfully complete an introductory programming course, they still find it difficult to design and code programming solutions.

This research investigates the effectiveness of using digital game *playing* to bring computer programming education into the world of experience of novice programming students enrolled in a college-level introductory computer programming course. The research determines whether or not digital game *playing* improves the effective transfer of the students' problem solving, critical thinking, logical, and programming knowledge from game playing to a formal programming environment. This research also explores whether this method is more effective for certain majors.

Acknowledgements

I am very fortunate to have had so many wonderful people touch my life in significant and enriching ways. Without them, I would never have been able to accomplish all that I have achieved. To each and every one of these people, I dedicate this work.

First and foremost, I thank my father, Francisco, and my mother, Lillian, for teaching me the value of hard work and education. By their example, I learned to strive to do my best in all things, to never fear trying something new, and to continually improve myself. They also taught me to love nature, literature, music, art, and playing games. Through the years, we spent many happy hours in libraries, museums, concerts, parks, beaches, and book stores, and we enjoyed endless hours playing games. They showed me the importance of maintaining balance in life.

I am forever thankful for my father's support throughout my life and especially through my undergraduate degrees. His never-ending support enabled me to embark upon my career in Information Systems (IS) which ultimately led to my career in academia. I could not have made it through those years without him. He is greatly missed.

I am grateful to my son Thomas for his continual support and encouragement through all of my four degrees. I could not have made it through any of them without you. I am so proud of all that you have accomplished in your personal as well as your professional life—you are a man of tremendous work ethic, integrity, compassion, intelligence, humor, wisdom, and are one of the most creative people I know. I value and respect your opinion and perspective, and love the time we spend discussing various topics. You share so many of my interests and passions, including playing games. Of all the people with whom I have played games, you remain one of my greatest challengers (and I love that).

Many thanks go to my grandson Jonathan. Prior to, and during this study, you helped me by testing the Scratch software in several different ways. You helped me to see Scratch from a young person's perspective and showed me that creating and modifying Scratch games was not difficult. Once I observed you using the software, I knew that my own students would also find it fun and easy to use. I love your creativity, artistic and intellectual abilities, and your sense of humor. I am so proud of you.

I am grateful to my sister Brenda. You have provided me with access to your home in PA which has afforded me with opportunities to rest, relax, and regroup. Having a quiet place to retreat to during these tough years was invaluable. Thank you for your support throughout my life and for attending so many of my graduations. I promise that this is the last one.

I loved computer programming from the very first programming course I attended. I will always be thankful to Prof. Wohl of Pace University for teaching me to program and for providing me with a strong foundation upon which I was able to build my IS and teaching careers. He was a great teacher.

I have so many wonderful friends who have been supportive of me throughout life's many ups and downs. A very special thank you to Millie Welkner, Ann Stevenson,

Denise Stevenson, Libby Britton, Teresa Ashley, Virginia Franklin, Sintia Molina, Maria Perper, Richard Giaquinto, Christy Scheese, Glenn Fehr, Jeanne Bergman, Pam Runge, Cheryl Howell, Starr Eddy, Rodney Roberts, Marsha Kane, Janine Stoyko, and the numerous other friends who have contributed so much to my life. Life would be empty without all of you.

The dissertation process would have been much more difficult had it not been for wonderful classmates and friends such as Rom Elizes, John Casarella, Larry Immohr, Dave Fronckowiak, Richard Washington, and Mirkeya Capellan. I learned so much from each of you, and I have the greatest respect and admiration for each of you and your many accomplishments.

I wish to thank my college for allowing me to conduct my research using the colleges' facilities, and Dr. Allen Burdowski for ensuring that I had two programming classes scheduled for each semester of this study. I am especially grateful to my students for their help in this research; I love teaching them and enjoy watching them learn. I wish to thank Christopher Schenone, a former student of mine, for creating several Scratch games and for allowing me to use them in this study. I also wish to thank Joyce Vogel for hiring me and for recognizing the value of my industry IS training and experience. You mentored me during my early days in academia and encouraged me to pursue both a master's and a doctoral degree. I am ever grateful to you.

I could not have gotten through the statistics in this dissertation without the assistance of Dr. Michelle Hirsch who taught me how to use the SPSS software and set up my input data, and Dr. Mark Koltko-Rivera who recommended several good statistics books and helped me to make sense out of some of my test results. I am grateful for your support and for your willingness to share your knowledge with me.

I love Pace University and I am grateful to all of the faculty members who taught us in the doctoral program. The university's doctoral program is innovative in that it allows working adults to pursue a doctoral degree (within a reasonable time frame) while maintaining full time jobs and family responsibilities. I wish to thank my advisor, Dr. Joe Bergin, for his guidance and expertise. It was a pleasure working with such a knowledgeable mentor, and I appreciate your patience, insights, and suggestions. I also wish to thank Dr. Fred Grossman. None of us would get through this program without your never-ending support and encouragement. And I will not soon forget the Socratic Method that you introduced us to in the doctoral process. I am thankful to Dr. Tappert. You introduced many of us to the world of biometrics and to the topic of futurism. I hope you will continue to send us those great articles even after we graduate. Lastly, I am so thankful to have had Dr. Mike Gargano as a professor twice—once as an undergraduate student and again as a doctoral student. He was a brilliant mathematician and professor, and was a genuinely nice and caring person. He is greatly missed by all of us.

Dr. Sandra Westcott

December 2008

Table of Contents

List of Tables

List of Figures

Chapter 1 Introduction

1.1 Problem Statement

Students preparing to enter the workforce need to have a wide variety of skills in order to be successful in the 21st century workplace. They need to know how to interpret and manage information and symbols from multiple sources, and know how to work with, and integrate, different media into their everyday tasks. Because students need the ability to work in teams and collaborate with others of diverse backgrounds, communication skills are critical. Students also need to be risk takers, know how to analyze and assess situations accurately, be creative problem solvers, and know how to make critical decisions quickly. They need the ability to multitask and handle change effectively. Today's workforce needs to possess a visual literacy that was not needed a few decades ago. [32, 41, 62, 70] Students can learn these needed skills in a variety of ways. One way students may acquire some of these skills is through playing today's complex digital games. Another way students might gain some of these skills is by learning computer programming.

Although computer programming may be effective in teaching students how to develop needed skills such as analytical, critical thinking, and problem solving skills, according to a survey conducted by the Higher Education Research Institute at the University of California at Los Angeles (HERI/UCLA), student interest in computer science as a major has declined by 70 percent since 2000 and by the fall of 2006, only 1.1

percent of incoming freshmen expressed any interest in computer science as a major. [80] While some might attribute this decline in enrollment to the widely-held public perception that most technical jobs are being outsourced overseas, a much more important reason for the decline in enrollment may lie in the lack of preparedness of many students to handle the rigors of the major and college-level work in general. If this country is to maintain a technological leadership role in this global economy, the decline in enrollment in computer majors needs to be reversed.

Students may not be interested in majoring in computer science courses but they do love technology and playing digital games. They have grown up using technology and playing digital games. Studies have shown that 98% of today's students have used personal computers and 96% of them have gone online for various reasons. [68] In addition, studies have indicated that by the time students have reached the age of 21, they will have played over 10,000 hours of digital games but will have spent less than 5,000 hours reading books. [10] Students today are different. Their needs are different. Their learning styles are different.

Attention needs to be given to the learning styles of today's novice programming students. Many of them have learning styles that differ from the styles of their programming instructors. Because these students are accustomed to playing exciting and engaging video games that contain vivid graphics and interesting plots and storylines, [34, 62, 66, 73], they may prefer to learn in a more sensory, immersive way. [60, 62] Students who are sensory learners may quickly become bored when they are required to learn new concepts by simply reading programming textbooks, listening to technical lectures, or writing algorithms and code. This may be particularly true when they do not

see the relevance or practicality of the topics and exercises to their lives or career goals. Because of the sensory learning styles of today's students, there are several researchers who believe that digital games may prove useful in motivating student learning. [2, 6, 8, 9, 23, 29, 34, 60, 63] The learning styles of students are a significant factor affecting the assessment outcomes of students enrolled in an introductory programming course. [77]

Today, college students of various disciplines are often required to take an introductory computer programming course. As the complexity of computer programming languages has increased over the years, computer science professors have noticed that novice programmers have had increasing difficulty learning to program. [35, 43, 64] Several multi-national, multi-institutional studies have shown that even after students successfully complete an introductory computer programming course, they cannot design and code a programming solution to common programming exercises.[25, 46, 53]

For years the main focus of many introductory programming courses has been on teaching novice programmers the programming language syntax. [3, 18, 20, 71, 74] This approach to programming presupposes that students entering an introductory programming course already possess basic proficiency in critical thinking, problem solving, and logic. In addition, the context of many programming exercises that are assigned to novice programmers are based in business concepts, mathematics, or computer science and thereby assumes that students have the basic knowledge needed to complete the programming exercises. [18, 51, 56] Perusing through the word problems contained in some programming textbooks reveals that this knowledge includes, but is not limited to, knowledge of profit margins, price mark-ups, taxes, metric conversions,

quadratic equations, price discounting, rational numbers, compounding of interest, Fibonacci numbers, and factorials. [30, 49, 69] Novice programming students not only need to learn the programming language, environment, terminology, concepts, logic, and debugging techniques, they also need to learn how business works and need to understand mathematical concepts they may not have previous learned or mastered. This places an enormous cognitive load on many novice programmers, and for students who may have no fundamental understanding of business, have a weak foundation in mathematics, are academic underachievers, or have weak critical thinking and problem solving skills, this introduction to computer programming may lead to frustration, a change in major, or failure.

When the main focus of an introductory programming course is on syntax, topic material is often presented to students all at once whether or not the students need that information to perform the assigned exercises. [14] For example, when students need to learn about a particular data type, the traditional approach is to teach the students about all data types that exist in the programming language being studied regardless of which data type is needed for the assignment. This overwhelms students with too much information and students tend to disregard information that they do not deem as necessary for the task at hand. [17, 61] However some researchers have found that for novice programming students learning the programming syntax is not as big of a problem as figuring out how to put "the pieces together" to create a complete, logical, working program. [31, 37, 64]

Several studies have found that novice programmers think differently than expert programmers. [22, 35, 43, 51, 81] According to Lister, expert programmers organize

information in more flexible, abstract ways. [47] They create mental schemas based upon experience and then are able to apply those schemas to similar situations. [56] Expert programmers also tend to think in algorithmic patterns as opposed to the line by line approach used by novice programmers to solve problems. When confronted with new programming problems, novice programmers do not see the similarities from previous problems and therefore, they tend to be context specific in their problem-solving approach. [43] Even after students have successfully completed an introductory programming course, numerous studies have shown that students are still unable to write a coherent computer program to solve a problem. [25, 36] In an attempt to remedy that situation, some educators have tried to engage novice programmers by having the students write computer games. [11, 39, 42, 44, 48, 75] This research differs from the game writing approach.

1.2 The Scope and Purpose of this Study

This study explores the effectiveness of using digital game *playing* in teaching computer programming to first-level programming students. While many studies have used game programming to teach introductory programming courses, no research has been found that has used digital game *playing* to teach programming concepts and constructs to novice programmers.

The goal of this research is to bring computer programming education into the world of the students' experience in a way that is engaging and meaningful to them. This research addresses the questions as to whether or not digital game *playing* can improve the learning assessment outcomes of novice programming students.

Digital games are constructed using the basic programming constructs (sequential, selection, and iterative) that students need to learn in a first-level programming course, and students tacitly learn these constructs through hours of repeated game playing. The research design developed for this study will use a series of steps to facilitate the conversion of the students' tacit knowledge of these constructs to explicit knowledge, thereby providing the foundation upon which the students can anchor C++ programming concepts. The research design has three main steps: black box game playing, white box game playing, and transition to the real-world C++ environment.

The research will begin by having students play Internet-based digital games they know such as Solitaire and Tetris. Because students will not be able to actually see the programming code for these games, the games are considered a black box game environment. Mapping the game playing to the programming constructs begins with the instructor explicitly discussing the rules of the game being played in terms of the programming concepts and constructs. For example, in the game Solitaire a player draws a card and needs to decide if that card should be placed on one of the four suit stacks according to suit and numeric sequence, whether the card should be placed on one of the seven row stacks according to suit color and numeric sequence, or whether the card should be placed in the discard pile. Students will explicitly be told that this is an example of the selection construct and that any time there is a decision point in which there are alternative choices of action based upon whether or not a condition is met, it is an example of the selection construct. (More detail about this mapping is provided in section 3.3.) After mapping several more examples of game rules to the selection construct, students will have the opportunity to provide additional examples from their

own every day life experiences which illustrate those construct discussed. This is important so that students understand that the concepts and constructs they are learning are not context specific. This mapping process will be repeated for each of the programming constructs.

The next step in the process is to transition students to a white box game environment where they can still play digital games, but now will have an opportunity to glimpse at the code behind the games they are playing. The Scratch software will be used as the white box game environment and will serve as a bridge between the game playing environment and the C++ programming environment. The software is a simple, easy-to-use, multimedia, color-coded, drag-and-drop game environment created by MIT Media Labs.

During this white box game environment step, not only will students play the games and see the code behind the games, they will also have the opportunity to modify the code to enhance the games they are playing. In order to modify the code, students will have to read the code, be able to understand the logic, and then fit their modifications into the existing logical design of the game. Students do not have the opportunity to modify or enhance the games they play in a black box game environment. The instructor will explicitly help students in mapping the programming concepts and constructs learned in the black box game environment to the corresponding programming concepts and constructs learned in the white box game environment so that students do not think that these constructs and concepts exist in specific contexts or environments. For example, when students are playing Diver II in Scratch, they will be able to see that if the shark is touching the diver, the score will be decremented by three points. They will be reminded

that this is an example of the selection construct and will be reminded of the selection construct examples discussed while playing games in the black box environment. They will be encouraged to discuss similar examples found in this game as well as other games. This process will help students to connect the black box game playing environment to the white box game playing environment and will help them to recognize that the construct exists across contexts. The mapping process will be repeated for each construct. (More detail about this mapping will be provided in section 3.3.)

The final step in the three-step mapping process is to transition the students from the white box environment of Scratch to the real-world C++ programming environment. Explicit classroom discussion, as well as course materials provided to students, will be used by the instructor to illustrate the mapping between the programming concepts and constructs learned Scratch and the corresponding concepts and constructs in the C++ programming environment. This mapping process will create the foundation upon which students can anchor the C++ concepts and constructs. For example, when learning the selection construct in C++, students will reexamine Scratch games such as Tetris to see how the construct was implemented in the Scratch game and how that implementation can relate to the C++ program they are trying to write. Students will also be reminded of similar selection construct examples from the black box game environment.

For each of the constructs, the instructor will help students to abstract the concepts from the white box and black box game environments and apply them to the real-world C++ programming environment. This will be done through continuously reminding students of what they already learned in each game environment and also through the prepared materials which provide a mapping between some Scratch exercises

and the C++ programming environment. (More detail about this mapping will be provided in section 3.3.)

Ultimately, the research results do conclude that using this digital game *playing* approach in a first-level programming course does in fact improve the overall learning assessment outcomes of novice programmers.

1.3 Dissertation Summary

The focus of this dissertation is on the effectiveness of using digital game playing to teach programming constructs and algorithmic patterns to novice programming students enrolled in a first-level college programming course. Both a Pilot Study (fall 2007) and a Formal Study (spring 2008) were conducted at a small, liberal arts college in Brooklyn, New York. The college is an ethnically diverse, commuter school. The student population is predominately first-generation college students with an average SAT score below 1000 on the combined verbal and math sections of the test. [57]

Compared to their counterparts, first-generation college students begin college at a disadvantage and appear to be less academically prepared for the demands of college course work than their peers. [40, 52, 76, 79] As a result, many first-generation college students begin college by taking remedial courses to make up for their academic deficiencies. [79] According to researchers, first-generation college students enter college with lower GPA's, math, reading and critical thinking skills. [40] In addition, because their parents did not attend college, first-generation college students have less access to information about what to expect during their college career. [79] Often they have unrealistic expectations of the demands of college life and course loads and unlike their