Using a Serious Game Approach to Teach ‘Operator Precedence’ to Introductory Programming Students

Nicoletta Adamo-Villani  
Department of Computer  
Graphics Technology  
Purdue University  
West Lafayette, IN, USA  
nadamovi@purdue.edu

Thomas Haley-Hermiz  
Department of Computer  
Graphics Technology  
Purdue University  
West Lafayette, IN, USA  
thaleyhe@purdue.edu

Robb Cutler  
Department of Computer Science  
Purdue University  
West Lafayette, IN, USA  
rcutler@purdue.edu

Abstract

In this paper we describe the design, development, formative evaluation and initial findings of one level of a serious game whose objective is to teach Information Assurance concepts to undergraduate students in introductory programming courses. The game level focuses on the concept of ‘operator precedence’. The player travels through a multilevel 3-dimensional maze and at each junction in the maze he/she is required to solve a mathematical problem that involves the application of operator precedence rules. A correct answer allows the player to move closer to the maze exit, an incorrect solution moves the player farther from the end of the maze. Initial findings from a formative study with a group of 14 undergraduate students show that the game level is usable, engaging and useful for learning/reviewing the intended programming concept.

Keywords: Serious Games; Information Assurance; Computer Science Education; Usability Evaluation

I. INTRODUCTION

Computer security breaches seem to be a daily occurrence in the news. New versions of computer viruses, worms, and Trojan Horses abound. As just one example of the problem, by the end of 2011, over 1.7 million computers were infected with the Conflicker worm allowing remote access of the computer without the user’s knowledge. The problem is so bad that operating systems vendors such as Apple and Microsoft release patches as often as weekly in order to fix security problems that could compromise a user’s computer.

The field of information assurance grew out of the need to formally manage and control such risks related to information technology, specifically in the use, processing, transmission, and storage of data. The size and scope of this problem can be summarized by a 2010 report that found that 58% of software is susceptible to security attacks [9]. One measure of the growing importance of information assurance education is seen by the promulgation by the National Security Council of a set of standards that a university must follow to be designated as a Center for Academic Excellence in Information Assurance Education (CAE/IAE). Unfortunately, by the end of 2009, only 3% of all universities had received that designation, most of whom focus their study of information assurance education at the graduate level [10].

While certainly not all security breaches are due to poor software engineering practices, programming errors are a well-known and major contributor to software security issues. Yet many programmers and software engineers do not have advanced degrees and have received limited formal training in information assurance.

In order to help address the need to teach introductory undergraduate programmers information assurance concepts, we describe the design and development of one level of a serious game whose purpose is to teach information assurance concepts to undergraduate students in introductory programming courses. We show the results from an initial formative study where we tested the usability, engagement, and efficacy of the game play with undergraduate students.

II. BACKGROUND

A. IA education and importance of the operator precedence concept

Information assurance education is typically taught to upper-level undergraduates or to graduate students. Yet given the number of programmers who have little knowledge of these concepts and the vast numbers of software applications with security issues, the importance of teaching information assurance concepts earlier in the computer science curriculum seems clear. To do so, many challenges must be overcome to reach these beginning programmers.

First, while information assurance concepts are easy to describe on a superficial level, presenting an in-depth look at information assurance is much more difficult. Furthermore, students often cannot gain a thorough understanding of information assurance concepts without sufficient programming experience – something not present in beginning programmers.

Second, the curriculum in an introductory programming course must cover both extensive programming language syntax as well as a wide variety of basic computer science algorithms and data structures. Simply put, there is little room in the syllabus for additional topics.

Third, the goal of most introductory programming classes is to teach programming, not software engineering. Testing and code review, when it is done at all, is often only performed as an afterthought. Ensuring that a program also has no security errors is beyond the scope of most students in an introductory class.

Finally, the instructor often has little understanding of information assurance and thus provides little, if any, instruction in the appropriate information assurance concepts in programming.

Operator precedence is a programming term for the mathematical concept known as the order of operations. Given an expression built from a set of operators and a set of operands, evaluation of the expression can have different
values, depending on the order in which the operations in the expression are performed. For example, the expression $6 + 12 \div 3$ has the value 10 if the expression is evaluated using standard mathematical rules for the order of operations (do division before addition), but has the value 6 if the expression is evaluated by performing the operations left-to-right without regard for any operator precedence.

In programming as in mathematics, an expression is evaluated according to a set of operator precedence rules. Unfortunately, the rules may be different from one programming language to another or may be unintuitive to the programmer. In C or Java, the example expression above would evaluate to 10 as one would expect mathematically. However, in Smalltalk, the expression would evaluate to 6 because Smalltalk evaluates all expressions in strict left-to-right order.

In languages with implied multiplication, the rules for operator precedence may be more complex. When x is 2, the expression, $10/5 \times x$, could be evaluated as either 1 or 4, depending on the order of operations.

Mixing unary and binary operators can cause confusion as well. Many programming languages evaluate the expression $-4^2$ as $-16$, performing the exponentiation before the negation. Yet a leading spreadsheet application performs the negation first, giving a result of 16.

These examples have been simplified to only use a few operators in order to illustrate the ambiguities that can occur with operator precedence. A modern programming language such as C or Java may have as many as 15 different precedence levels for over 50 operators.

Programmers who do not understand the concept of operator precedence or are not conversant with the precedence levels are likely to write code which may give incorrect results at best or create security holes at worst.

B. Serious Games

In Mike Zyda’s definition, a game is “a physical or mental contest, played according to specific rules, with the goal of amusing or rewarding the participant”, while a serious game is “a mental contest, played with a computer in accordance with specific rules that use entertainment to further government or corporate training, education, health, public policy, and strategic communication objectives” [19].

Serious games are designed with the intention of improving some specific aspect of learning. Play, an important contributor to human development, maturation, and learning, is the main component of serious games [5].

The founders of the Education Arcade at MIT stated that there are many intrinsic motivations for learning associated with games. “The threat of failure is lowered. Games allow players to try, make mistakes or fail, and then try again without losing face. Discovery and application of learned skills in new contexts encourages exploration and experimentation. A sense of engagement continues during gaming. Computer games allow players to be stakeholders in the events that occur on the screen” [8].

Some research has taken place on serious games from a "new media" [6], psychological [12][7], and sociological perspectives [14]. As far as learning, empirical evidence to support the assumption that serious games are educationally effective is still limited. Serious games may have positive impacts as education tools [17], but relatively little is known about the cognition of the users who play them [16]. Blunt [1] demonstrated that students taking business, economics, and management courses that had an additional serious game component performed better than students that took the course with- out the serious game component. Wong, et al. [18] found that non-science major students taught physiology using a serious game methodology performed better than those taught using traditional static text and images. Collard and Scott [2] and Collard and Shernoff [3] taught a numerical methods course centred completely on a serious game in which students wrote programs that would race a virtual car around a track. They discovered that students in this course spent more time out of class working on the material, demonstrated deeper learning, greater intellectual intensity, more intrinsic motivation, and increased engagement as compared to traditional approaches. Rosas, et al. [13] found that the use of games in the classroom, via portable handheld devices, led to improved motivation and learning for math and reading in primary school. In a study performed by Papastergiou [11], results showed that a computer game based learning approach was effective in fostering students’ knowledge of computer memory concepts.

Cooper, Dann, and Pausch [4] incorporated Alice game environment into a course for programmers with little or no programming background. Results of the study showed that students who took the Alice-based course performed better in a Computer Science level one course (CS1) than the total group tested, and significantly better than the students with similar backgrounds, who did not take the Alice course.

III. THE GAME

The setting of the level is a technologically advanced three-dimensional maze. The maze has three floors; higher floors are reached with three consecutive correct answers. Each floor has a different texture and look (Figure 1). To win the game the player must get to the third floor and successfully answer three problems in a row. All problems require the application of various operator precedence rules. If a player gets a total of three wrong answers, he/she loses and must start the maze over. As the player navigates the maze, at each impasse, a robot (Figure 2) aids the player by presenting an operation and two outcomes. By selecting the correct outcome the player advances in the correct direction in the maze. The maze is built dynamically each time the game is played, therefore, hindering the player from simply remembering the correct direction and allowing for repetition of play.

C. Technical implementation

The game was built using Unity3D, an object-based game development environment that uses a Javascript-like language for scripting the objects in a scene.

As the user progresses through the maze, expressions are generated on-the-fly from a pre-determined set of operators and set of operands. Attributes of an operator include its type (binary or unary), associativity (left or right), precedence (a value relative to other precedence values), and an evaluation function (to compute a value when the operator is applied to its operands). Attributes of an operand include its value and an array holding its fully parenthesized representation. Both operators and operands
also have a visual representation used when the expression is displayed to the user in the maze.

When an expression is presented to the user, two answer choices are also given. One answer is the correct evaluation of the expression; the other is an incorrect, yet plausible, evaluation of the expression. Plausibility is defined as an answer that could be computed through incorrect application of operator precedence rules. Thus two different expression evaluation functions are implemented.

Because operands and operators are designed as abstract data types with both visual and semantic representations, the game can work with a wide variety of concrete data types and their precedence-based operations. Examples include integers (1, 2, 3, …) and their associated arithmetic operations (+, –, •, ÷), and boolean values (true, false) and their associated logical operations (and, or, not), and sets (…, –). The game is being created using an iterative user-centered development approach that includes two forms of evaluation: formative and summative. Formative evaluation focuses on the design features of the game (i.e., usability, fun and engagement and quality of the graphics); summative evaluation tests the efficacy of using the serious game for teaching various information assurance concepts to undergraduate students in introductory programming courses.

In this paper we describe an initial formative evaluation of one level of the game with a group of undergraduate students. The goal of the study was to answer the following questions: (1) is the game level usable, engaging and visually pleasing, (2) do students find the game useful for learning/reviewing the concept of operator precedence, (3) how can the game level be improved. In addition, the study also aimed to determine whether gender and video game experience have a significant effect on the ability to play the game.

A. Study Design

The study collected quantitative and qualitative data. Quantitative data included time spent on the game activity, number of game-play mistakes, number of correct/incorrect answers, and completion/non completion of the game level. Quantitative data also included students’ answers to a web survey with rating questions pertaining to the usability and visual quality of the game levels. Answers were based on a 5-point Likert scale ranging from strongly agree to strongly disagree.

Qualitative data included ‘think aloud protocol’ and ‘critical incidents’, i.e. problems encountered that affect task flow and performance, and answers to open-ended questions. In addition, all testing sessions were screen captured and video recorded. The footage was scored with reference to a set of positive and negative instantiations. Positive instantiations that were looked for were smiles, laughing, signs of excitement, and positive vocalization. Negative instantiations were frowns, signs of boredom, signs of frustration, and negative vocalization.

This study was a quasi-experiment, where the sample population was pre-selected. Subjects were recruited randomly from classes and programs at Purdue University as well as Indiana University Purdue University at Indianapolis (IUPUI). Participation in the study was voluntary.

B. Subjects

Fourteen (14) undergraduate students age 19-25 enrolled in Computer Science and Computer Graphics Technology. 6/14 subjects were females.

C. Procedure

Testing was performed in a controlled lab setting. Each participant was directed to sit in front of a laptop with an external mouse device attached; a video camera was set up near the laptop focusing on the participant’s face. Each subject began the test by answering demographic questions via an online survey. Upon completion, the participant was asked to read the online instructions pertaining to the level of the game; then he/she was instructed to open the executable of the game level and play. Participants were allowed to play the level as long as they pleased. After completion or failure of the level, each subject was asked to refer back to the online survey to answer questions about the level he/she had just played. After all questions for both levels were answered, the participant was asked questions about the overall experience.

D. Findings

Game metrics. The mean time spent on the level was 02:42 (minutes: seconds); the maximum time spent on the level was 06:23; the minimum was 01:34. All subjects were able to complete the level. The error rate average was 14.8%.

Answers to rating questions. Overall, participants found the level fairly engaging (Mean= 1.9; strongly agree=1 and strongly disagree=5). Participants’ reaction to the quality of
graphics (i.e. color scheme, look of the environment and robot, and animations) was positive (mean = 1.8; strongly agree=1 and strongly disagree=5).

Answers to open-ended questions. Answers to open ended questions show that subjects felt comfortable with the game controls. 5/14 subjects complained about the game environment being too dark and the text being difficult to read. Responses regarding the most fun aspect of the level predominately pointed to the graphics. 10/14 subjects responded that they would play the level again. 3 subjects commented that the game “…is a really cool way to learn and review the operator precedence rules”.

Observation, video recordings and screen captures. Observation and video recordings showed that the majority of the participants were engaged during game play. Screen captures revealed that the game interface was found very easy to use. Only 3/14 subjects invoked the help menu during game play; 11/14 subjects were able to complete the game level without any help.

Effect of gender and video game experience. A two-sample hypothesis test was performed on the metrics to determine if gender and video game experience were statistically significant. Results show that gender and video game experience did not have a significant effect on the time spent playing the level, the ability to complete the level, and the error rate. However, some of the female participants found the look of the maze to be not “female-friendly” and suggested to “add a few maze levels with more feminine color and texture treatments”.

V. CONCLUSIONS AND FUTURE WORK

In this paper we have described the development and initial evaluation of one level of a serious game whose goal is to teach information assurance concepts to introductory programming students. The level presented in the paper focuses on operator precedence rules in programming. Findings from a study with 14 undergraduate students show that the game is usable, fairly engaging and useful for learning/reviewing the concept of operator precedence.

Future work will include completion of the seven game levels and assessment of learning outcomes. Summative evaluation will be conducted once the game is completed to: (1) assess the overall worth and effectiveness of the game; (2) draw out key lessons learned from the project; and (3) determine the sustainability, transferability, scalability, and relative importance of the initiative in enhancing students’ understanding of information assurance concepts as well as increase students’ retention in computer science majors.

ACKNOWLEDGEMENTS

This research is supported by the National Science Foundation (NSF) - CCLI program, award # 0622900 (Aug. 2010 - Dec. 2013).

REFERENCES


[9] Krauss, L. et. al., 2010. 58 percent of software vulnerable to security breaches similar to Google, Department of Defense cyber attacks. [online] Available at: <http://www.veracode.com/content/view/1015/38/>


