A Games-Based Approach for Teaching the Introductory Programming Course

Rathika Rajaravivarma
Dept. of Computer Science
Central Connecticut State University
New Britain, Connecticut 06050 USA
RajaravivarmaR@ccsu.edu

Abstract

Introductory programming courses in computer science aim at building an effective foundation for the development of programming skills. A prudent way to develop these skills is by emphasizing problem solving and logical thinking. This paper proposes a games-based approach, as a way of engaging students and developing these skills. Common mistakes of novice programmers in traditional courses are summarized. Word and number games are used to illustrate the potential benefits of a games-based approach, which minimizes such mistakes.

Keywords: CS1, pedagogy, games-based approach, game programming, word & number games, active learning

1. Introduction

Innovative pedagogical approaches to teach the introductory programming course, often referred as CS1, is an ongoing topic of discussion across universities and colleges around the world. The course is centered on the three aspects of programming - design, development, and testing. Inadequate balance in applying these concepts results in disproportionate amount of time spent by the students, leading to frustration and lack of motivation. This is in fact a serious problem in this gateway course to computer science. Motivation and involvement are crucial factors in retaining students in a specific program. In this paper the author emphasizes the problem identification aspect of problem solving and proposes a games-based approach that will bring a balance to the different phases of program development. This approach encourages experimentation and keeps the students involved and engaged.

In the next section, the difficulties students encounter and the mistakes they make are summarized. The significance of the games-based approach, which provides an active learning environment, is discussed in the subsequent section. Here how this approach helps in minimizing the usual weaknesses encountered in the traditional approach is discussed. Two classes of games – word and number – are described in the fourth section, where several individual games in these classes are identified for enhancing the learning of specific programming skills. Additional benefits of the games-approach and its intrinsic value in motivating and engaging the students are discussed in the concluding section.

2. Shortcomings in the Program Development Lifecycle using the Traditional Approach

The only way to learn programming is by actually doing it. In an effort to accomplish this effectively several approaches have been suggested [2,6,8]. In simple terms, programming is problem solving using a computer language. Once the problem is identified in the analysis stage, programming involves three main phases: design, development, and testing. This section discusses the commonly observed weaknesses encountered, in each of the four stages, with the traditional approach for teaching CS1.

2.1 Analysis or Problem Identification

To identify the problem, conventionally the problem that must be solved is specified to the students in a descriptive form. Sometimes the description of the problem is supplemented with classroom discussion. In spite of this, many students have trouble interpreting the instructions [2]. They misread the information or miss the important details or many times they do not know what the outcome they are looking for.

2.2 Design

The next stage is designing the algorithm to solve the problem on hand. When given a project, the immediate reaction for majority of the students is to start coding,

skipping the design phase. In spite of discussing the software development activities and requiring the use of design tools and detailed program description, many students relate projects to coding. They fill in the documentation and the design diagrams after the completing the code. The phase of designing a solution to the problem is lost in this approach.

2.3 Implementation

In the implementation part of the project, it is not uncommon to see students start working on the code without knowing where to start or what to do. Even when they do know what to do, they are lost when they encounter a problem. In an attempt to fix the code they tend to change a few syntax and statements randomly, leading to more confusion. In some instances they seek for help right away without making an attempt to think through the problem and the solution [7]

2.4 Testing

In the testing stage, the students tend to turn in the outputs for which their program works well. The range of input and the critical decision points are missed many times. When a program does not function the way it is supposed to, the students, due to lack of motivation, give up and fail to work through the problem and correct the errors.

3. Significance of the Games-Based Approach

In order to minimize the above weaknesses in the traditional approach, an alternate games-based approach is proposed in this section.

3.1 The Games-Based Approach

Involving the students in critical thinking and applying these skills to problem solving is a task by itself. In an attempt to accentuate this concept, some institutions offer courses focused on problem solving. The approach in this paper uses games to supplement programming with the logics of critical thinking. In this approach, the students explore the problem in the form of games. They figure out the problem themselves by playing the games. Each class has a scheduled 10 minutes at the end of the period allotted for playing games. The games played could be individual games or team games. The games played are not limited to the programming projects assigned to the students. Games such as brainteasers, puzzles, mathematical problems, or quiz type games can be introduced to make the students open up for out-of-the-box thinking practice. In the following class meeting time, the students turn in their solution to the problem using a design tool. They write down the steps to execute the game and the process, they think will be needed, to implement the solution. The games assigned for the projects are discussed in the following class meeting time. This allows the students to think through the problem or play the game enough number of times to get the hang of it.

3.2 Characteristics of Game-based Approach

By analyzing and solving both simple and complex games, the students create a stimulating environment that reflects their learning. The games-based approach challenges the students and enhances their problem-solving capabilities. Dealing with the challenges and coming up with appropriate solutions provide an enriched active learning environment. The brain-based learning theory [5] refers to this active learning process as metacognition: *how you know what you know*. The learners in this approach are encouraged to explore, gather and use information for some of the commonly played games. Also, working in teams to solve complex problems promotes collaborative learning and strengthens the interpersonal communication skills.

The uniqueness of the game-based approach comes with the involvement and the excitement of the accomplishment. This method enables each student to spend his or her own time to figure out the problem. They explore different ways of arriving at the solution. In simple fun games, the students may repeat the process just because they want to see a different outcome. This keeps them engaged and involved in the process. The game-based approach acts as a confidence booster. A positive stimulating experience is created when the computer games they developed are shared with or executed by their friends and families.

3.3 Minimizing the Weaknesses Using The Gaming Strategy

Problem solving through games underscores the problem identification and the design phases. Playing games provides a visual representation to the abstract nature of the problem. Conveying and describing the abstract concepts is a difficult task in teaching programming. This difficulty is overcome by game playing. By requiring the students figure out the problem by playing games, the problem identification phase is enhanced. Viewing the outcome and interpreting the problem directly from the outcome clears the misinterpretation. The design phase is materialized when the students tackle the problem and propose a solution using design tools. In simple games this phase is not so apparent. But games that require a strategy to solve (or need an algorithm) emphasize design.

The games-based approach enables the student to tie in the design with coding. In some instances, the students will not be able to start coding until a solution is figured out. The visual outcome of game playing defines the goal and serves as the starting point for the program development phase. The solution for the game may not be so obvious and there may be different ways to arrive at the solution. Creative solutions are a great way to motivate and involve students. It also gives them realities check on how sound their solutions function. When the program developed does not satisfy the required logics, it has the potential to turn on the self-directed motivation. The

students tend not to give up until they get the desired results through numerous tests.

4. Illustration through Two Classes of Games

The course starts off with simple games that utilize basic programming skills. As the course progresses games that requires additional programming skills are implemented. While simple games specific to the topic of discussion give a head-start for novice programmers, more complex games such as Nim [4] and Hangman [6] can be implemented in phases. It should also be noted that the projects and programs for the CS1 course may not be limited to the games mentioned here. The traditional programming projects can be supplemented wherever such project is more appropriate to the topic of discussion.

The games discussed in this paper are grouped into two broad categories: word games and number games. Word games focuses on string and character manipulation and number games focuses on working with integers and random numbers. Both types of games take input from users and read data from input files. Many of the games discussed here are two player games, played against the computer and the user. Since the approach to introduce Java programming varies with institution [1,3], the choice of using a GUI approach or terminal-based approach is left to the discretion of the instructor.

4.1 Word Games

Five different word games are discussed here. Many of these games start with a simple version and extend to complex versions requiring additional functionalities. These variations are implemented in several stages. The final project is the implementation of Hangman. Table 1 lists the different word games and their variations. It also summarizes the programming skills used in each game.

Table 1 List of Word games

Word Games and description	Stages of implementation	Programming skills needed
Interactive User Input - Input data from user to insert words in an existing passage. 1. *Crazy passage* (~ mad-libs) 2. *What is …?* (~ jeopardy)	1. passage such as telephone answering message can be used to create a crazy message. 2. a. get answers from user for a passage like word definition b. check answers c. compute number and percentage of correct answers	- Terminal Input/Output - String Concatenation - Control Structures - Use of algorithms to compute statistics
Cryptogram – Create a secret code from a given word or text. Crypt and decrypt the message	1. A simple character translation 2. translation from x characters to y characters 3. translation based on an algorithm	- Terminal Input/Output - String / Character Manipulation - Use of simple algorithms
Word Search – From a given line(s) of text, search for a specific word.	1. The line of text to be searched from and the word to be searched for can be input by the user or read from a file. 2. Search across multiple rows for data from a file.	- Terminal / File I/O - String Manipulation - Command Line Arguments - Parameter Passing - Arrays
Puzzle Maker – Use word(s) entered by user to create a text for word search.	1. Create a fixed line of text with random characters before and after the word to be searched for. 2. Repeat above for multiple words 3. Repeat above to create multiple lines of text	- Terminal / File I/O - String / Char. Manipulation - Random Numbers - Command Line Arguments - Control Structures - Arrays
Hangman – In this game, the user has certain number of chances to guess the secret word displayed in a cryptic form. If the user picks the letters in the word before the chances run out the user wins.	1. Generate the crypt for the word read from file. 2. Create the different stages of Hangman. 3. Display alphabet list and available alphabet list. 4. Put together stages 1 – 3	- Command Line Arguments - Terminal / File I/O - Random Numbers - Control structures - String / Char. Manipulation - Parameter Passing - Arrays - Exception Handling

4.2 Number Games

Five number games suitable for novice programmers are discussed here. The first game starts with the player guessing a randomly generated number. The randomly generated number could be generated by the computer and guessed by the user or narrowed by the computer using a suitable logic and a series of questions leading to the answer. Random numbers are used extensively in working with number games. This provides lots of variations to the problem that the programmer needs to think ahead. Finally, the game of Nim, where players remove objects from two stacks alternatively, is implemented. A sample screen shot of the terminal-based version of this game is shown in Figure 1.

In order to win the game of Nim, programmer has to come up with a strategy. This part of the game emphasizes the design process. A winning logic must be created before the implementation. The different number games and the programming skill sets needed to implement the games are listed in Table 2.

5. Conclusion

Unlike other projects implemented in the introductory programming course, the game-based approach provides a better possibility to reuse and extend the capabilities of the code developed in this course. This approach will be utilized for the CS1 course in the forthcoming semester and the effectiveness of this concept will be published. The games-based approach underlies the importance of working with algorithms and can be extended to advanced programming and data structures.

Sharing the games, which were created by students themselves during the span of this course, with friends and families creates a positive environment for the students and a sense of ownership in designing a solution to the problem. More importantly, this approach creates a passion to want to do more, a crucial factor sought after in an introductory programming course. Further, this approach stimulates the need to learn more advanced features that could be applied to the gaming environment in creating games that are more fun and exciting.

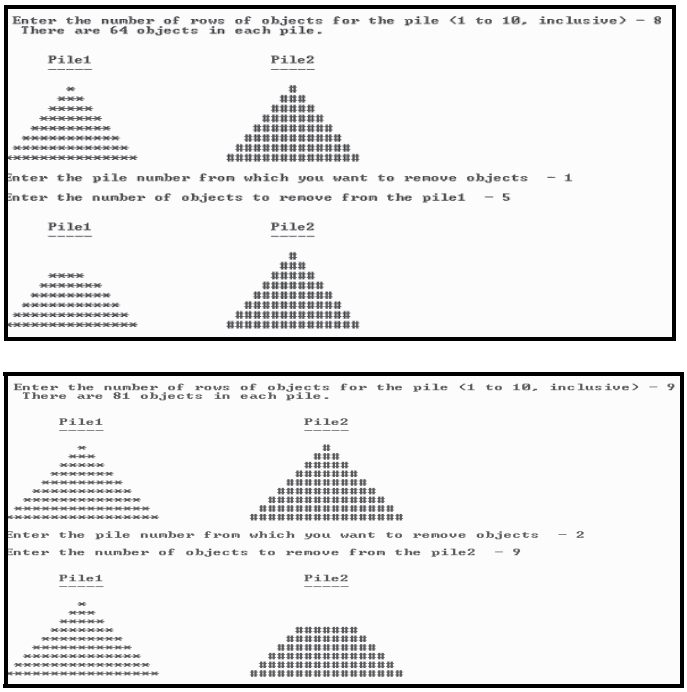

Figure 1 Sample screen shots for terminal-based Nim

References

[1] Barnes, D., and Kölling, M., *Objects First with Java - A Practical Introduction Using BlueJ*, Prentice-Hall, 2002.

[2] Giguette, Ray, "Pre-Games: Games Designed to Introduce CS1 and CS2 Programming Assignments," Proceedings of the 34th SIGCSE Technical Symposium on Computer Science Education, Vol. 35, No. 1, 2003

[3] Goldman, K. J., "A Concepts-First Introduction to Computer Science," Proceedings of the 35th.SIGCSE Technical Symposium on Computer Science Education, Norfolk, VA, 2004

[4] Greco, J., "Designing a Computer to Play Nim: A Mini-Capstone Project in Digital Design I, Proceedings of the ASEE Annual conference and Exposition, Salt lake city, UT, 2004

[5] Ladeau, A, Brain-based Learning, http://www.fcae.nova.edu/~turgeonm/bbl.html

[6] Meyer, J. and Dwyer, C., "A Case Study In Teaching Programming Using A Hybrid Instructional Model," The Proceedings of ISECON, v 17, 2000

[7] Pollard, S. and Forbes, J., "Hands-On Labs Without Computers."Proceedings of the 34th SIGCSE Technical Symposium on Computer Science Education, Vol. 35, No. 1, 2003

[8] Raviv, Daniel, "Hands-On Activities For Innovative Problem Solving," Proceedings of the ASEE Annual conference and Exposition, Salt lake city, UT, 2004.

Table 2 List of Number games

Number Game and Description	Programming skills used
Guess the Number – 1. Use of simple algorithm to compute the result. Perform reverse calculations wherever applicable. 2. The Hi-Lo guessing game User guesses the number generated by the computer. Computer provides hints as to high or low guess until the user guesses the correct number. 3. The computer guesses the number the user had in mind (like birth date, month, or year) using series of (yes / no) questions	- Terminal Input/Output - Random Numbers - Control Structures - Emphasis on program design (Version 3)
Slot Machine – Generate 3 different random numbers. If they match, claim that the user is winner; else provide appropriate messages.	- Random Numbers - Control Structures
Rock, Paper, Scissors – A simple game of strategy with a specific winning combination - With rock and scissors - rock wins; With scissors and paper - scissors win; With paper and rock - paper wins. The program can offer the choice of the computer or the user starting the game. At the end of each game winner is announced. The game is repeated until the user desires.	- Terminal Input/Output - Random Numbers - Control Structures - Emphasis on program design
Coin / Dice / Card games – A number of different games can be generated using one or more computer simulated coin(s) / dice / card(s).	- Terminal Input/Output - Random Numbers - Control Structures - Used defined class(es) - Emphasis on program design
Game of Nim – This is a game based on strategy. From a stack of one or more piles players alternate by taking one or more of the objects from a pile. The player taking the last object or stack of objects is the winner. Can be generated in the terminal-based or GUI based form.	- Terminal Input/Output - Random Numbers - Control Structures - Emphasis on program design - Arrays - Passing arrays of objects