Reilly, C.F.; De La Mora, N., "The impact of real-world topic labs on student performance in CS1," *Frontiers in Education Conference (FIE), 2012* , vol., no., pp.1,6, 3-6 Oct. 2012

DOI: 10.1109/FIE.2012.6462329

The published version of this paper is available from IEEE Explore:
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6462329&isnumber=6462204

The Impact of Real–World Topic Labs on Student Performance in CS1

Christine F. Reilly† and Noe De La Mora
Computer Science Department
University of Texas – Pan American
1201 University Drive
Edinburg, Texas 78539
†Corresponding author. Email: reillycf@utpa.edu

***Abstract*—We examine the impact of using lab exercises based on real–world topics in the CS1 course at the University of Texas – Pan American. In Fall 2010 and Spring 2011 we used drill style exercises. For Fall 2011 and Spring 2012 we created a new set of lab exercises that are based on real-world problems. In this paper we examine impact of the new lab exercises on the number of students who complete the exercises, on the students' grades on the exams, and on the final course grade. For the new lab exercises, the students are provided with an example program that contains extensive comments describing the skills targeted in that lab. Then they complete a similar program on their own. Whenever possible, we used games for the programs. When we could not devise a game exercise, we used problems that the students are likely to encounter in the real world. Sometimes we reused the same game/problem in multiple exercises. We found that many more students completed the lab exercises and the overall course performance improved when we used the new labs.**

I. Introduction

The University of Texas – Pan American (UTPA) is a Hispanic serving institution located in South Texas. There are over 19,000 students enrolled at UTPA, with 88% of those students being Hispanic, and 68% of the students being first generation college students. The university as a whole is pursuing student retention initiatives in order to increase the percentage of students who graduate in six years, which now stands at 42%.

In 2011 (spring and fall combined) 13 Computer Science and 15 Computer Engineering students graduated from UTPA. Our Engineering Computer Science I (CS1) course, which focuses on introductory programming in C++, is required for both the Computer Science and Computer Engineering majors. Each section of the course usually has between 25 and 35 students enrolled. In the fall semester we usually offer three sections of CS1, and we usually offer two sections in the spring semester. We note the significant difference in the number of students entering the majors with the number graduating. There are non-majors who take CS1, but there are usually only a few per section because we have a separate introduction to programming course for non–majors.

Like many other schools, our introductory computer science courses have a high fail/drop rate and we are actively seeking solutions for this problem. We observed that because the laboratory portion of the course contributes to 30% of the course grade, those students who were not completing many lab exercises were less likely to succeed in the class. Additionally, the students who do not complete lab exercises do not get the programming practice provided by these exercises and may have poorer performance on the more complex programming assignments and exams.

In this paper we present an initiative to increase participation in the laboratory component of the course by creating lab exercises that focus on topics that we think would engage student interest. We used games for the lab exercises as much as possible, and used other real–world problems for the remaining exercises. Although we focus our study on the CS1 course, we encourage others to study the impact of using real–world topic assignments in other computer science courses, and in engineering courses in general.

This paper is organized as follows. In Section II we describe other research that is related to our work and that provided the inspiration for our study. We then describe the lab exercises we designed in Section III. The comparison of student performance in the course when we used the original labs versus when we used the new labs is presented in Section IV. Finally, we conclude our study and discuss avenues for future work in Section V.

II. Related Work

It has been widely acknowledged that using game–themed assignments in introductory programming classes helps to engage students. When using games in introductory programming classes, students should be learning computer science concepts through the use of games, as opposed to learning how to program games or about game related algorithms [1]. Some of the prior research has examined the use of video–games as CS1 assignments, while others have used board games. The research generally acknowledges that graphical games are better than text–based games in engaging students, but that the graphical games can require significant development time on the instructor's behalf. A major motivation for using game–themed assignments is that students can easily relate to the topic [2].

Drake and Sung [3] examined the use of board games for programming assignments in CS1 and CS2. They used well known games such as tic-tac-toe, hangman, and backgammon,

TABLE I
COMPONENTS OF CS1 COURSE GRADE

Component	Weight
Class Participation	10%
Programming Assignments	20%
Exams	40%
Lab Exercises	30%

as well as more obscure games, including European board games that are gaining in popularity in the United States. They provide suggestions for the right type of game to use as a programming assignment. The game should have simple, but not trivial, rules. Games where the boards form complex graphs or that involve many different pieces should be avoided. The game should take five to fifteen minutes to play, and it is best to use games that require two players. Because both players will be sharing the same computer screen, games that require hidden information, such as a hand of cards, should be avoided. All of the games presented in their paper can be implemented using a text–only interface, but the authors acknowledge that students generally prefer a graphical user interface. They provide a Java library that supports both text–based and GUI programs. They utilized the board game assignments in a CS2 course at a small liberal arts college and received many positive comments about these assignments on the student evaluations.

While GUI–based games are thought to be more engaging to students than text–based console games, it is also acknowledged that GUI–based games require extra effort and knowledge from the faculty member. Some libraries have been developed for GUI–based games in Java [3] and in C# [1]. There is an introductory C++ textbook that uses games and graphics [4]. Most of the assignments in this textbook that we could use for our CS1 course are ones where the students write programs that draw rudimentary graphics on the screen and do not involve interactive play.

III. DESCRIPTION OF LAB EXERCISES

The CS1 course at UTPA has a co-requisite lab course. The students attend lab one day per week for two hours and forty minutes. The students' task during the lab class is to complete programming exercises. There are teaching assistants on hand to describe the programming exercises and to assist the students while they work on the exercises.

For grading, we treat the three credit lecture course and one credit lab course as a single four credit course by assigning a student the same grade in both courses. The grade is calculated as a combination of the exams, major programming assignments, class participation, and the labs, as shown in Table I. Because the labs count as 30% of the course grade, they contribute to a major portion of a student's grade.

A. Original Lab Exercises

In Fall 2010 and Spring 2011 we assigned lab exercises that are formatted as drills where the students are given certain skills to practice. The students are given a C++ program to complete. The program contains a demonstration of the types of skills students will use to complete the lab. For the most part, these lab exercises do not have a specific topic or goal for the program. The purpose of the programs is for the students to practice and demonstrate the specified skills. All of the description and documentation for the lab exercise is contained within the provided code skeleton. Some of the example code in the exercises contains skills that the students have not yet learned. Note that the original lab exercises were not written by the instructor who was using them. The instructor who wrote the labs may have taught topics in a different order, or briefly introduced more advanced topics earlier in the semester.

The students completed a total of 24 lab exercises, usually two per week. We provide some examples of these exercises, chosen for the purpose of comparison between these original and the new lab exercises:

- **Lab 1 (week 1) - Hello World and simple math:** The demonstration code first uses basic console I/O to interact with the user. Then, based on user input regarding the student's projected course grade, it uses if-else statements to print the corresponding letter grade. Note that our students had not yet been exposed to decision making statements. Finally, the lab demonstrates basic addition and then asks the student to write code that performs subtraction, multiplication, and division.
- **Lab 3 (week 2) - Type conversion, named constants:** The demonstration code shows integer division, casting, overflow, and underflow. Then the students are asked to use a named constant to compute the area of a circle with a user supplied radius. The code supplied to the students uses a function to generate a menu where the user chooses which operation they would like to demonstrate, if statements to perform the chosen operation, and uses an infinite for loop to continue the program until the user chooses to quit. At this point in the semester, we had not covered the use of functions, decision making, or looping.
- **Lab 6 (week 3) - if statements:** In this lab, the student is asked to input their score in the course, and the code uses if statements to determine the corresponding letter grade. The lab consists of three steps: using only if statements, using if–else statements, and using nested if–else statements. For each step, the student is provided with example code that determines the score that corresponds with the grades A and B, and the student completes the code for the remaining letter grades.
- **Lab 7 (week 4) - switch statement vs. if–else:** In this lab, the student is provided with code that asks the user to input a letter between a and d, and shows how if–else statements can be used to print a different phrase for each letter that the user could enter. The student is then asked to re–write the code using switch statements with an integer condition that selects what to print. Then they are asked to re–write that switch statement as a series of if–else statements. Note that the example code uses an infinite while loop to keep repeating the user

selection until the user chooses to quit (at which point it executes a break statement). At this point in the semester the students had not yet learned about looping.

- **Lab 14 (week 7) - Functions:** The example code for this lab demonstrates writing and calling both value returning and void functions, with and without parameters. The functions take the sum of two numbers, sometimes obtaining the numbers as parameters and other times retrieving the numbers from user input. The students are then asked to write similar functions that ask the user for their first and last name, then print the name.

B. New Lab Exercises

For Fall 2011 and Spring 2012 we implemented new lab exercises [5] where the students were asked to use particular skills to solve a given problem. Our goal when designing these labs was to make them something that the students could relate to and that they would be more likely to be interested in completing. Whenever possible we used games for the programming exercises. When we were not able to devise a programming exercise using a game, we focused the exercise on a problem that the students are likely to encounter during their daily lives.

The lab exercises are all console–based programs. We chose this format because we are not familiar with graphical programming in C++. We wanted to quickly develop new lab exercises so stayed with the format we are familiar with. As discussed in Section II, it has been shown that using graphical programming assignments engages student interest and we will consider exploring the use of graphical programs in the future.

All of the labs have a similar format. The lab is described on a web page that contains links to example programs and the program skeleton for the lab exercise. First, the students are given an example program to review. The example program demonstrates the skills the students are practicing in this lab and contains extensive comments describing these skills as they are encountered in the example program. Then the students are asked to complete their own program. For most labs, they are given a skeleton of the program with preliminary code and comments that guide them on how to complete the lab. They are also provided with sample output from the correctly completed lab exercise.

The students completed a total of 21 exercises. In the early part of the semester, they completed two or three exercises per week. From the middle of the semester onward, an individual lab exercise was generally more complex than those from earlier in the semester and the students completed one exercise per week. We have chosen a sample of the lab exercises to present in this paper, generally those that provide a good illustration of our design or those that are a good comparison with the original lab exercises.

- **Lab 1 (week 1) - First C++ program:** The purpose of this lab is to teach the students the basic principles of using Visual C++ (the IDE we use) and running a simple "Hello World" type program. The example program is "Hello World", and the students complete a program where they print their name. The only C++ skills required for this lab are basic statements and console output.
- **Lab 2 (week 1) - Basic Debugging:** In this lab, the students are provided with a version of the "Hello World" program that will not compile. They must identify and correct the missing quotation marks around the output text and insert a missing semicolon. The description for this lab shows students how to interpret the compile error messages, how to use the compile error message to find where the error is in the program, and to address each compile error one at a time.
- **Lab 3 (week 2) - Basic Math:** This lab adds basic math skills (addition, multiplication, division, and modulus) to the skills covered in previous labs. The example program uses an equation to calculate the "Dateable Age" for the user (the youngest age they "should" date based on the user's age) using a formula that involves addition and integer division. The instructions point out that dividing two integers results in an integer, which is the correct result in this case. The students then complete a program that calculates a "magic number" using a formula that involves multiplication and modulus.
- **Lab 4 (week 2) - Console I/O and random numbers:** This is the first lab were we found games suitable for the content to be demonstrated. The example program simulates a coin toss by generating a random number between 0 and 1. The student's program is the beginning of the development of the Pig dice game [6], a game they will continue to develop as the semester progresses. In this lab they ask the user to enter his or her name, roll the die once (generate a random number between 1 and 6), and print the value that was rolled. In this lab the students use a random number generation function that is provided in the lab skeleton code, and use the getline function from the string library. The use of pre–defined functions had also been discussed in class at this point of the semester.
- **Lab 6 (week 3) - if statements:** The example program for the if statements lab simulates a drive–thru restaurant where the customer chooses their menu item by number. Based on the number entered by the customer, the program uses if–else statements to determine the amount to charge for the order. The program that the students complete simulates the Magic 8-Ball toy (a fortune telling toy). The students must generate a random number then use if–else statements to pick the corresponding answer from the Magic 8-Ball. Note that nested if statements are covered in a separate lab.
- **Lab 9 (week 4) - switch structure:** The example program and student program are the same topic as the if statements lab, but now the selections are made using a switch structure. The instructions encourage the students to compare the switch solution with the solution from the if statements lab.
- **Lab 15 (week 7) - Functions:** The example program asks the user to input the price of an item in two consecutive

years and calculates the inflation rate for that item. It uses functions for user input, calculating the inflation rate, and printing a number with two decimal points followed by a percent sign. The students' program continues to develop the game of Pig [6] that we had started in earlier labs. The students are provided with a program that plays the Pig game for a single player. Their task is to create a two–player version of the game. The students must write a value returning function that gets user input (checking for correct input), a value returning function to play a single turn of the game, and a void function to print the output in a specified format. They must also complete the main function by calling the functions that they write.

C. Discussion of Original and New Labs

Both the original and new labs provide demonstration code then ask the students to complete a program. The new labs provide a more complete description of the task at hand, and generally focus solely on a single new skill. Some of the examples in the original labs contain code that the students had not yet learned in class, while the new labs restrict the code in both the example and student's program to code that has been covered in class. One benefit of the original labs is that they are very complete in covering the various skills and often directly identify these skills. The new labs do not cover the skills in such detail, but do discuss the skills that are covered in the comments in the example programs.

IV. Comparison of Student Performance

We compared the performance of students in four semesters of CS1 that were taught by the same instructor. In Fall 2010 and Spring 2011, the students completed the original lab exercises. In Fall 2011 and Spring 2012 the students completed the new lab exercises.

A. Lab Completion Rate

The first metric examined was the total percent of labs that were completed in each semester. This number was derived by taking the sum of the number of labs that were completed and dividing that sum by the number of labs that were assigned. As shown in Fig.1, in Fall 2010, 71% of labs were completed, in Spring 2011, 66% of labs were completed, in Fall 2011, 82% of labs were completed, and in Spring 2012, students completed 81% of the labs.

There is a dramatic increase in the number of labs completed when the new lab exercises were used. We take this result as an encouraging sign that students might be more engaged by the new lab exercises and therefore decide to complete more of the exercises. Another possible reason for the increase in lab completion is that the instructor emphasized the impact of the labs on the course grade (labs contribute 30% to the course grade) more in Fall 2011 and Spring 2012 than she did in Fall 2010. However, she did also emphasize the importance of the labs to the students in the Spring 2011 class, which had the lowest lab completion rate.

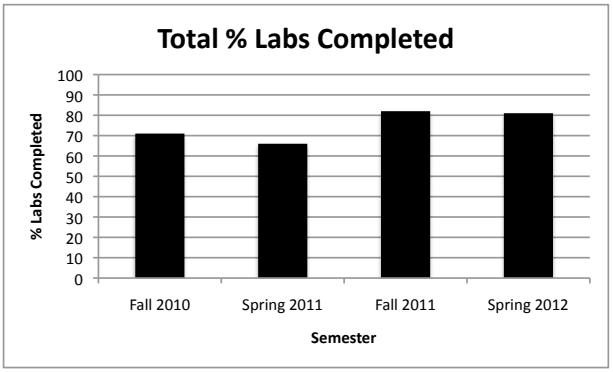

Fig. 1. Lab Completion Rate per Semester

B. Exam Grades

We are also interested in examining if the new labs had any impact on the students' exam grades. It is possible that by completing more of the lab exercises and, therefore, getting more programming practice, the students will perform better on the exams. Fig.2 shows the grades on the three exams for each of the four semesters. Each set of bars in this figure represents the grade on one of the exams for each of the four semesters. The average grades are presented both excluding and including zeros. As is especially evident in Spring 2011, there are students who stop participating in the course and decide to accept a grade of F instead of dropping the course. Because of changes in financial aid rules with regard to failing grades that were implemented in Fall 2011, we expect that fewer students will simply stop participating in the course in future semesters.

For this discussion, we focus on the average of exam grades excluding zeros because the students who received a grade of zero on the exam were most likely also not participating in lab. For all three exams, Fall 2010 has the lowest average (61, 67, and 65 for Exams 1, 2, and 3, respectively). Spring 2011 has the highest average score of 80 for Exam 1, but has the second lowest scores for Exams 2 and 3, of 71 and 75, respectively. The highest grades on Exams 2 and 3 were achieved in Spring 2012, with averages of 79 and 78, respectively. In Spring 2012, the average for Exam 1 was 75. In Fall 2011 had the average score was 77 for all three exams.

With the exception of Exam 1, the students in Spring 2012 performed better on the exams than students in the other three semesters, followed by the students in Fall 2011. This supports our hypothesis that getting more programming practice through completing more lab exercises leads the students to perform better on the exams. However, in the absence of a controlled study we are unable to confirm this hypothesis.

C. Course Grades

Completing more lab exercises has a direct impact on a student's grade in the course because the lab exercises contribute to 30% of the course grade. Most students receive a high grade on each lab exercise, with many receiving grades of 100% on each exercise. Therefore, we expect that by

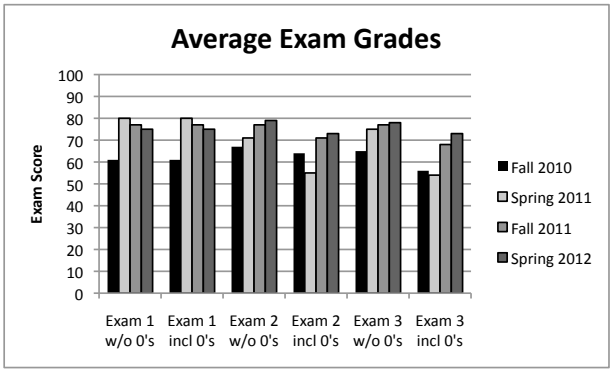

Fig. 2. Average Exam Grade per Semester

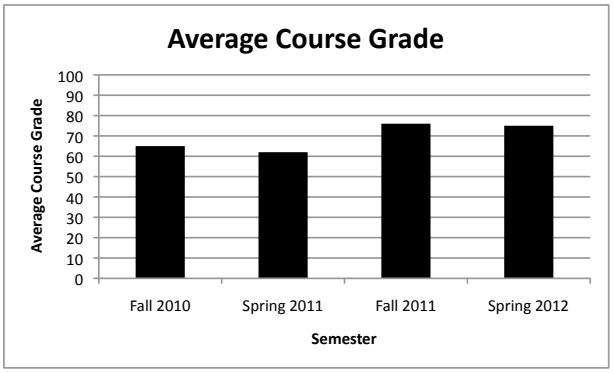

Fig. 3. Average Course Grade per Semester

completing more lab exercises, students will earn a higher grade in the course. This expectation is confirmed by our data, shown in Fig.3. Students in Fall 2011 had the highest average course grade of 76, followed by Spring 2012 with an average of 75. Fall 2010 and Spring 2011 had the lowest course grades of 65 and 62, respectively.

D. Pass Rate

We examined the proportion of students in each semester that passed the course. Students must receive a grade of C or higher in order for CS1 to count in the Computer Science or Computer Engineering curriculum. Fig.4 shows the pass rate of students in each semester, with the left side bar for each semester representing the percent of students who received passing grades at the end of the semester, and the right side bar including those students who dropped the course in the number of students who failed the course.

Our results show that the students in Fall 2011 and Spring 2012, when the new labs were assigned, had a higher pass rate than those in the previous two semesters. By completing more labs the students directly increase their course grade, as discussed above in Section IV-C, leading to more students passing the course when there are more students completing the labs.

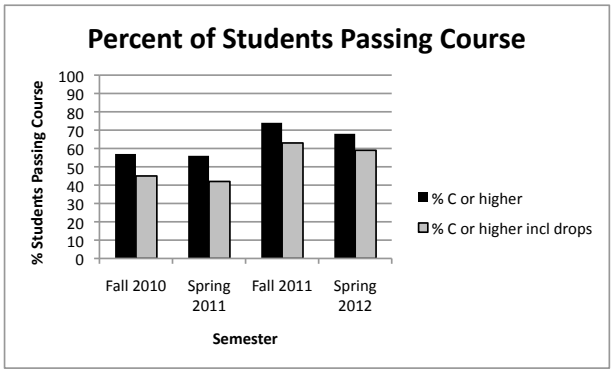

Fig. 4. Pass Rate per Semester

V. Conclusions and Future Work

Our conclusion is that the students perform better in the course when they complete the game/real world problem based lab exercises. However, we cannot draw a correlation between implementing the new lab exercises and the students' overall performance in the course. One reason for the higher course grade is that by completing more of the lab exercises, the students are directly increasing their grade. The fact that the students in Fall 2011 and Spring 2012 performed better on all but the first exam may indicate that more programming practice results in better retention of the material. Another factor that may affect the students' performance that the instructor did not have much teaching experience prior to Fall 2010. Despite the fact that we cannot draw a correlation between the new lab exercises and the students' course performance we are encouraged by the increased participation in the lab and by the large increase in the course grades.

As of the time this paper was written, we had not yet received the student comments from the student course evaluations for Fall 2011 or Spring 2012. Therefore we are unable to report on student opinions of the new lab exercises. Other sections of CS1 courses at UTPA, including the version for non–majors, utilized some or all of the new lab exercises. Feedback from the professors who taught these courses was very positive. We will continue to monitor student performance in CS1 when these new labs are used to determine if the improved student performance will be seen in future semesters.

We plan to continue refining the new lab exercises. In the future, we would like to explore the possibility of implementing exercises in C++ that involve a graphical user interface. We would also like to work on replacing the non–game topic exercises with game–based exercises.

REFERENCES

[1] K. Sung, M. Panitz, S. Wallace, R. Anderson, and J. Nordlinger, "Game–themed programming assignments: The faculty perspective," in *SIGCSE 2008*, 2008.

[2] J. D. Bayliss and S. Strout, "Games as a "flavor" of cs1," in *SIGCSE 2006*, 2006.

[3] P. Drake and K. Sung, "Teaching introductory programming with popular board games," in *SIGCSE 2011*, 2011.

[4] T. Gaddis, *Starting Out with Games & Graphics in C++*. Addison-Wesley, 2010.

[5] "CSCI/CMPE 1170.03/1370.03 labs. utpa.edu http://faculty.utpa.edu/reillycf/courses/labs1370/. accessed: April 15, 2012."

[6] "The game of pig. gettysburg.edu. http://cs.gettysburg.edu/projects/pig/index.html. accessed: April 15, 2012."