Back to the Basics. In an Effort to Improve Student Retention in Intro to Programming Classes

Ed Lindoo, Ph.D
Regis University
3333 Regis Blvd.
Denver, CO 80221-1099
303-964-6358
elindoo@regis.edu

ABSTRACT

High failure and drop-out rates in introductory programming courses are a big concern for institutions, instructors and student's. After reviewing the literature presented within this paper, it became apparent that the problem is not merely a local challenge at our institution (with a 50+ percent drop/fail rate), but rather a universal phenomenon. Student performances in these intro courses show a large variation across the student population for a variety of reasons such as teaching methods, teacher competence, the student's self-discipline but most importantly student motivation. In his study, Alturki [1] investigated motivation using a two-factor theory, where factors causing satisfaction (motivators) and dissatisfaction (hygiene factors) were applied to students in his courses. The results showed that students with good intrinsic motivation had less difficulty completing the course than other students whereas problems arose with students who had poor motivation.

The Alturki [1] study concluded that low motivation is a major problem that affects pass rate. Three steps were introduced to increase motivation. 1) reduce de-motivators, 2) increase intrinsic motivators by making assignments and projects more appealing to students (ie: visualization and real-world projects) and 3) introduce intrinsic motivators by asking students to submit sections of assignments weekly rather than all at once. After these improvements, the pass rate increased from 44% to 68%. A´lvarez (2016) also determined that motivation is directly related to student satisfaction, and what could satisfy students more than not only learning, but passing a course with a good, if not exceptional grade. For these reasons, it is proposed to take a more simplistic, more motivating approach in teaching intro to computers by teaching students a simple, basic programming language called Xojo, while at the same time assigning real-world projects and giving extra credit assignments to keep scores, and therefore motivation high.

INTRODUCTION

Students success or failure in the introductory programming course often determines their majoring in computer science and related fields. If a student drops out, fails, or passes with difficulty, Malik [8] found that it is unlikely that the student will enroll in a follow-on course. Despite extensive research on factors that influence success of new students in introductory programming, it is still not fully understood what makes an introductory programming course positive and successful. However, Alturki [1] found that low success rates in introductory programming were attributed to many factors, including student motivation, the intrinsic difficulty of programming and the complexity of course structure. Suggestions to overcome these problems and improve success rates include using less complex course structure, employing more collaborative learning, redesigning course material, and improving teaching style. Alturki

[1] also pointed out that researchers have found failure is not always associated with cognitive ability, but often with motivation and teaching style.

Previous studies have found a strong connection between student performance and their motivation. For example, Corney et al. [5] redesigned the CS1 introductory programming course to be more engaging and collaborative to motivate students. In addition to using paired-programming students during lab sessions, they redesigned the course to provide more information and engaging content about various technologies like databases, the Web, and networking. The goal was to make programming more interesting and more relevant to real-world problems. The changes resulted in higher success rates, less dropouts and a course geared to what industry requires from graduates (to be good communicators and team players who are business-minded). According to Corney [5], the feedback on these changes from their students showed the importance of making programming relevant to real problems. Therefore, the focus of this paper is to find ways to motivate students in and introduction to programming course.

MOTIVATIONAL AND DEMOTIVATIONAL FACTORS

The biggest challenge in learning programming is to acquire different sets of skills at the same time [1]. Beginning students need to learn both syntax and semantics of a programming language, while at the same time develop problem solving skills. To compound this challenge, programming courses require students to study theoretical concepts and practice these concepts while designing and developing programs. Therefore, novice programmers must learn multiple concepts and apply these in a practical manner concurrently, often pushing students into overload [8]. Interestingly, Pillay & Jugoo [10] found that students who had taken mathematics and problem-solving prior to their CS1 course had a significant and positive improvement in CS1. Pillay and Jugoo [10] also found course set-up and the choice of programming language an important issue, one which plays a considerable role in the way programming concepts are introduced to students.

Alturki [1] pointed out that there have been many debates about the most suitable language to use for an introductory course. He found that most institutions are starting students out in Java, C, C++, Python or Ruby. However, Barland [3] a professor at Radford argues that C/C++ and Java are poor choices to learn as a first language. As Barland [3] points out, "Programming is a difficult task, learned over months and years. Object-oriented programming (the "++" part of "C++") is a more advanced topic which is important for larger programs, but is best taught after the fundamentals have been learned." Barland [3] went on to say that "teachers know it is only common sense not to distract from a topic by teaching advanced details to a beginner, yet that is exactly what's happening if we throw C++ or Java at them." Malik [8] also pointed out that 30 % of their programming course students dropped out because they found the programming exercises too hard and difficult. Backing the comments of Barland [3], Malik [8] went on to say that the usual approach in teaching programming is to start with the syntax of a programming language (usually Java or a flavor of C) and move on to the associated semantics. This has proven to not be the best approach as motivation can quickly drop.

During the early 1980's, the bulk of schools teaching programming were doing so with languages such as COBOL, Basic/Qbasic, Fortran, Pascal, RPG and assembler. Of these, Basic was arguably the most disliked because it was very unstructured, not object oriented and the ease of writing poor programs was exponential. Although the C language had been developed in the 70's, it wasn't until later in the 80's when mini computers took off that C was introduced into the college curriculums. When it was introduced, it took off, quickly jumping to the forefront of teaching methods. However, as previously discussed, C is not a very intuitive language to learn [3].

On the other hand, Pseudocode, an artificial and informal language is very easy to follow and helps programmers develop algorithms. Pseudocode is “text-based” (algorithmic) and written in English as a detailed yet readable description of what a computer program or algorithm should do [9]. It is expressed in a formally-styled natural language rather than in a programming language. For example, table 1 below presents Pseudocode on the left and Basic programming code on the right.

Pseudocode	Basic Code
Initialize total to zero	Dim total as integer = 0
Initialize counter to zero	Dim counter as integer = 0
Entering 999 will end the program	Dim average as double
Input the first grade	Input “Enter Grade zero to 100”, grade
while the user has not entered 999	While grade <> 999
if a valid grade is entered	If grade => 0 and grade < 101 then
add this grade into the running total	total = total + grade
add one to the grade counter	counter = counter + 1
input the next grade (possibly 999 to end)	Input “Enter Grade zero to 100”, grade
if the counter is not equal to zero	If count > 0 then
set the average to the total divided by	Average = total/counter
the counter	print “Average = “; average
print the average	end if
else	else
print 'no grades were entered'	print “No grades were entered”
	end if
	Wend ‘this is the end of the while loop

Table 1

Reading through the Pseudocode on the left is quite easy and intuitive, and with just a little Basic programming training, it goes without saying that Pseudocode and Basic code are very similar. As Mohd Rum [9] points out, Pseudocode makes creating programs easier, allowing programmers to concentrate on the logic of the problem without having to know programming language syntax. Pseudocode depicts the entire algorithm’s logic so that it requires only rote implementation to translate it line by line into executable source code. Again, this simplicity is demonstrated in table 1.

PROPOSAL

Given the research presented, it is proposed to go back to the basics, Basic programming that is. Not back to the days of Basic or Qbasic, but to the present day of an object-oriented programming environment called Xojo, which uses Basic as the underlying code. As shown in figure1, Xojo (pronounced Zo-Jo) is, on the surface a drag and drop programming environment. With Xojo, many different types of apps can be built, including web-based apps, console (or command line) apps and iOS apps. The view shown in figure1 is called the Layout view in which the user visually designs the look and feel of the application. Behind the layout view is the code view, where Xojo source code is entered to control the behavior and functionality of the application.

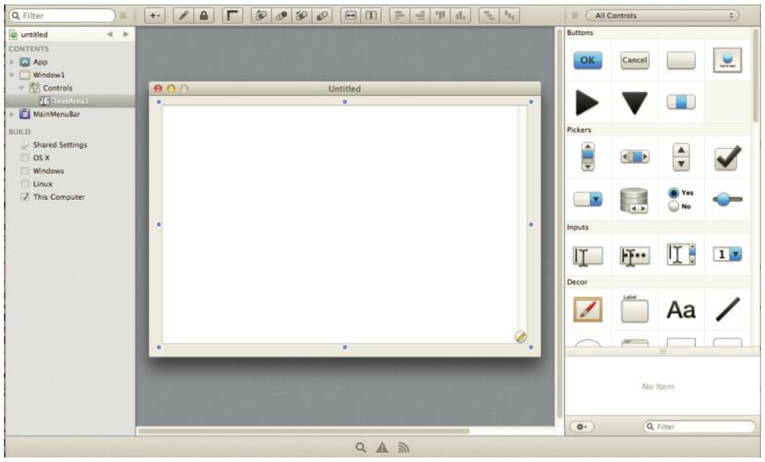
Figure 1

As components are added to projects, they will be displayed in the Navigator, whether in Layout View or Code View. Double clicking on an item will open it for editing. These can be arranged in any order desired by a simple drag and drop. The order in which these items are arranged has no bearing on the performance or functionality of the application. It is up to the designer to organize your project in a way that makes sense. The environment of XoJo is very much like Microsoft Visual Studio, however the full functionality suite (less the ability to create .exe files) is free from Xojo. And just like Visual Studio, code view is simple basic coding like this:

```
Dim counter As Integer
For counter = 1 To 10
    MsgBox(Str(counter))
Next
```

One of the obvious benefits of Xojo is the motivational value it brings because of ease of use and the ability to quickly and easily build a simple application. Gratification can be realized quickly without a lot of frustration that typically goes with programming. But Xojo also has a lot of fun things that can be built. For example, figure2 shows a program that not only stores information about customers, but also shows on a map where they live. In figure3, Xojo simulates an Apple iPhone giving the designer the ability to create applications for iPhone which when completed can easily be published to the Apple store.

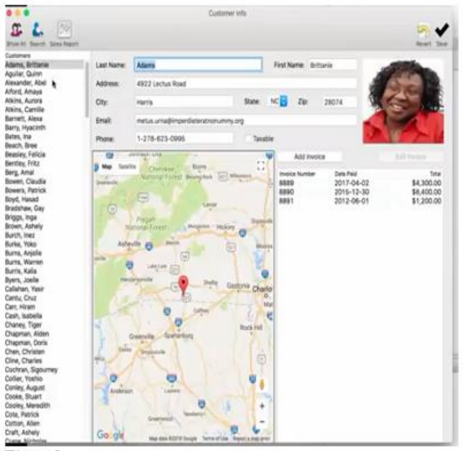
Figure2

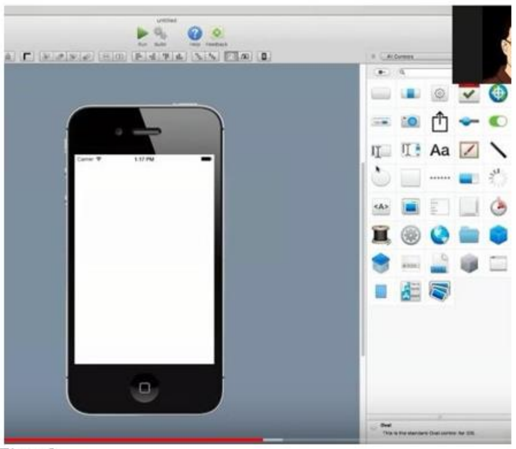
Figure3

A free manual, written by Xojo staff, not only walks the user through building a number of applications, but also covers many of the programming requirements of Basic. For example, Chapter three covers, If... Then, Select Case, For... Next, Do... Loop, While... Wend, Exit and Continue, and Hands On With Loops. Other chapters describe essentials such as Functions, Scope, Lists & Array’s, Object Oriented Programming, working with Files, working with Images, Printing, Networking, sending Emails, accessing Databases, Classes, as well as controls such as Buttons, Pickers, and List Boxes. In addition, Xojo also has an extensive library of hands-on videos (at: http://developer.xojo.com/videos) as well as example applications that come free when Xojo is downloaded. One of the side benefits of Xojo which can be very beneficial for business majors is that BASIC (VBA) is also used in the background programming of Excel and MS-Access. It should be noted that Xojo runs on Windows, Mac and Linux platforms.

CONCLUSION

Simplifying the course structure and reducing its complexity have proven to have a positive effect on student performance, motivation and retention [1]. By reducing complexity, students focus more on the fundamental concepts in programming. By learning in the Xojo environment, students don’t get bogged down in low-level programming languages that should be introduced later in a CS curriculum. As Carvalho [4] pointed out, one of the most challenging aspects of a computer science curriculum involves helping students learn the concepts of programming. However, A´lvarez [2] believed that the biggest problem novice programmers run into is their lack of program solving skills and because of this he points to high drop-out and failure rates in programming courses. However, to overcome these problems, A´lvarez [2] found that students need tools that help them to obtain these problem-solving skills. Practical learning situations are the most useful for learning programming, believes A´lvarez [2], and it is for this reason that he, and others advocate the use of visual programming environments which reduce the cognitive requirement to start working on programming tasks. Although visual environments do not solve the problems with the syntax, A´lvarez [2] believes that they could postpone the problem as they allow the students to firstly focus on the task, not all the syntactic rules.

Another area to look at and one not previously discussed is that of project-based learning (PBL). de-la-Fuente-Valentín, et al. [6] presented PBL techniques as an effective way to increase student motivation and reduce dropout rates. While there is extensive research literature that describes the use of PBL techniques within programming curriculums, it should be noted that it is widely used in modern education across many disciplines. The Educators of America [7] discuss the effective use of PBL as a “combination of collaboration, reflection, and individual decision-making gives the students an applicable scenario to real-world situations that they will face as they mature.” It is believed [7] that PBL allows students to collaborate in teams in ways to address or solve the given problem. Solving real-world problems is key to student motivation and de-la-Fuente-Valentín, et al. [6] back this in their discussion of understanding and solving open-ended problems in a collaborative manner. Further evaluation and trials in the PBL are recommended.

Finally, starting the fall term 2018, we will introduce Xojo at our university in an attempt to bring our 50+% drop/fail rate way down. It is anticipated that this rate will be cut in half (or better), to under 25% by using the Xojo environment, reducing de-motivational factors, increasing motivation, and using real-world projects.

References

[1] Alturki, R. Measuring and improving student performance in an introductory programming course. Informatics in Education, Vilnius University Vol. 15, No. 2, 183–204, 2016

[2] A´lvarez, A. and Larran˜aga. Experiences incorporating lego mindstorms robots in the basic programming syllabus: Lessons learned. Springer Science+Business Media Dordrecht, 2016

[3] Barland, I. Why C and C++ are awful programming languages. https://www.radford.edu/ibarland/Manifestoes/whyC++isBad.shtml, (2017)

[4] Carvalho. E. S. Analyzing the quality of student's interaction in a distance learning object-oriented programming discipline. Interdisciplinary Journal of e-Skills and Life Long Learning, 11, 85-99. Retrieved from http://www.ijello.org/Volume11/IJELLv11p085-099Carvalho0919.pdf, (2015)

[5] Corney, M., Teague, D., Thomas, R.N. Engaging students in programming. In: Twelfth Australasian Conference on Computing Education – Volume 103. 63–72. (2010)

[6] de-la-Fuente-Valentín, L and Pardo, A. and Delgado-Kloos, C. Addressing drop-out and sustained effort issues with large practical groups using an automated delivery and assessment system. Department of Telematics Engineering, University Carlos III of Madrid, Av. Universidad, 30, 28911 Leganés (Madrid), Spain (2012)

[7] Educators of America. What is project based learning? Buffalo, NY. https://www.educatorsusa.org/ (2018)

[8] Malik, S. and Coldwell-Neilson, J. A model for teaching an introductory programming course using ADRI. Springer Science+Business Media New York, (2016)

[9] Mohd Rum, S. N., & Ismail, M. A. Metocognitive support accelerates computer assisted learning for novice programmers. Educational Technology & Society, 20 (3), 170–181, (2017)

[10] Pillay, N., Jugoo, V.R. An investigation into student characteristics affecting novice programming performance. ACM SIGCSE Bulletin. DOI: 10.1145/1113847.1113888, (2005)