International Journal for Educational Media and Technology
2015, Vol.9, No. 1, pp. 56-67

Addressing Programming Anxiety among Non-Computer Science Distance Learners: A UPOU Case Study

Roberto B. Figueroa Jr
UP Open University
Laguna, Philippines
rfigueroa@upou.edu.ph

Emely M. Amoloza
UP Open University
Laguna, Philippines
emely.amoloza@upou.edu.ph

Studies have shown that anxiety has a negative effect on learning programming, which determines the success of students in a programming course, thereby decreasing retention rates. A worldwide campaign has been recently launched to search for ways and design learning activities for lessening or eliminating this anxiety among Computer Science learners. This study hopes to contribute to this effort by finding out if using innovative online interactive platforms in an introductory programming course will reduce programming anxiety among non-computer science students. Three online interactive platforms were utilized in an introductory programming course feared by many multimedia students at the University of the Philippines Open University (UPOU). Specifically, the new course design involved students under the program of Bachelor of Arts in Multimedia Studies taking MMS 141 during the third trimester of Academic Year 2013-2014. The students were observed and interviewed throughout the duration of the course and were asked to answer a 6-item questionnaire for programming anxiety that was patterned after the computer anxiety scale. Results have shown that there was a significant decrease in programming anxiety among these students after taking the course that employed these online interactive coding platforms.

Keywords: *Programming Anxiety, Open Education Resource, Distance Learning, Online Learning*

Introduction

Many students have the notion that computer programming and computer science as a whole is challenging. This is confirmed by a number of studies including that of Owolabi, Olanipekun, and Iwerima (2014) who argued that *"Computer programming which forms the major component of most technological inventions is perceived by most computer science students as difficult"* (p. 109), and Connolly (2009) who said, *"For some computing students, learning programming is intimidating, giving rise to lack of confidence and anxiety"* (p. 52). The phenomenon is called programming anxiety. Jenkins (2002) enumerated the different factors that bring about programming anxiety. They can be summarized into three themes:

1. **Multiple skills and processes.** Rogerson and Scott mentioned that, *"there are several barriers to learning programming, which include the nature or complexity of programming itself"* (2010, p. 150). Jenkins further elaborated that programming is composed of several skills that allow an individual to learn syntax, semantics, structure, and style. These concepts are unfortunately taught hierarchically and not holistically. Syntax is taught first, followed by semantics, structure, and finally style, which is sometimes even forgotten. This approach to learning is often reinforced by lectures and textbooks that concentrate on syntax details. The student, hoping to gain an understanding of programming, usually opts to read textbooks and listen to lectures instead of learning through experience. The processes involved in programming are requirement specification, algorithm design, and coding. The part that involves creating solutions for specified problems is usually ignored in traditional programming lectures; instead, special focus is given to coding. As a result of this teaching style, students learn to code, but not solve problems.

2. **Language and teaching style.** The choice of programming language was listed as a factor in affecting students' apprehension and discomfort. Programming languages for teaching programming like Pascal and Logo are usually replaced by the more industry-relevant and popular languages like C and Java. However, this choice of language adds to the stress of students as they try to learn the idiosyncrasies of such languages on top of learning the programming concepts, which should be the main focus of the course. In terms of teaching style, traditional examples like student records are given in boring lectures, worsening the experience of the students.

3. **Reputation and image.** According to Jenkins, programming courses have acquired the reputation of being difficult. This view is passed on by previous students, becoming more exaggerated with each telling, causing new students to expect the same level of difficulty as they try to learn programming. This idea is aggravated by the media's stereotypical representation of good programmers as socially challenged and inadequate nerds.

These may be the reasons why computer science has been found to be declining in popularity among undergraduate students since 2000 (Patterson, 2005). Almost suffering the same fate, teaching introductory programming and its basic principles to non-computer science majors has proven to be difficult, if not impossible, in a distance learning environment as many of the students tend to have an initial anxiety or aversion to programming. What makes it worse is the negative experience that many of them have in taking the course, which further cripples their confidence and interest in handling programming-related courses and tracks or degrees.

It is ironic that the country needs more graduates who are competent programmers. Unfortunately, although a lot of computer programming and IT-related degrees are being offered by many institutions in the Philippines, many of the graduates produced are incompetent programmers who fail miserably in basic programming, keeping the industry in want of qualified software developers (Pinaroc, 2008). The United States of America experiences a similar issue, as reflected by the country's huge dependence on outsourcing software development jobs to China, India, and the Philippines. Such prompted the need to form an organization that encourages citizens to go into IT-related fields, especially software development and programming (Miller & Wortham, 2011). One of the organization's biggest projects is the "Hour of Code" campaign, which aims to help K-12 students learn the basic principles of computer science with ease. This campaign utilizes a visual programming interface called Blockly, which makes it easier for students to appreciate programming logic and concepts without being intimidated by the complex syntax of programming language.

Studies show that anxiety, a big factor in determining the success of students in a programming course, has a negative effect on learning programming, thereby decreasing retention rates. On the other hand,

perceived learning, a non-cognitive aspect of education, is believed to have a positive effect on retention rates as it boosts motivation among students (Jenkins, 2002).

Programming anxiety was also found to be prevalent among students of an introductory programming course called MMS 141: Principles of Programming, which is offered under the Bachelor of Arts in Multimedia Studies program at the University of the Philippines Open University (UPOU). This encouraged the researchers' regard in increasing the interest of non-computer science students in programming while trying to decrease their learning anxiety.

Amid these problems, campaigns to popularize computer science and programming among learners have started to populate the web. In 2012, CodeCademy proclaimed Codeyear and introduced a gamified curriculum for those who wanted to learn programming using its interactive platform. In 2013, Code.org launched the hour of code, which introduced learners from any background to a fun and easy way of learning programming, through block-based programming activities (Brooks & Lindgren, 2015). As of this writing, there have been 119,863,373 participants in the "Hour of Code" (Code.org Leaderboard, 2015). Furthermore, another article stated that, "as of January 2014, over 24 million users completed over 100 million exercises in CodeCademy (Kim, 2015, p. 22)". These astounding numbers gave an impression of the engaging nature of these platforms, which eventually led to their selection as tools for this particular study.

In this attempt, three online interactive platforms were utilized in MMS 141. The earlier part of the course integrated the Code.org platform, an online course with a graphical programming interface for creating applications. The latter part utilized CodeCademy, another interactive platform that reinforces programming concepts and problem solving while teaching Javascript. The third platform called Blockly is the programming interface used by Code.org but has its own separate application. This application was used for students to build their projects.

This study aims to find out the effects of using these online interactive platforms on programming anxiety and perceived learning among non-computer science distance learners of the UPOU. This can be seen as a case study as it was just observed in one offering of the MMS 141 course.

Research Design and Methods

The study involved the use of three interactive platforms, namely Blockly, Code.org, and CodeCademy. A description of each platform is given in this section.

Three platforms

Blockly

Blockly is an open source visual programming editor that lets users drag blocks to create programs. It is written by Neil Fraser, a software engineer at Google who made Blockly's documentation and source code available at https://code.google.com/p/Blockly/.

Blockly has drawn much attention since its deployment that it became the featured programming interface for Code.org, whose creators launched the "Hour of Code". Another notable application is the MIT App Inventor, which allows programming novices to create their own Android applications by dragging blocks together.

The Blockly website also has an application for translating Blockly puzzles or code into industry-recognized languages such as Javascript, Python, Dart, and XML.

Code.org

Code.org is a non-profit organization whose main goal is to encourage appreciation of and participation in computer science by making it available in as many schools as possible ("About Us," n.d.). A consequential goal is to increase the contribution of women and minor ethnic groups. Code.org was founded by twins, Hadi and Ali Partovi, who were the angel investors of Dropbox and Facebook (Delevett, n.d.).

The organization believes that "computer science and computer programming should be part of the core curriculum in education, alongside other science, technology, engineering, and mathematics (STEM) courses such as biology, physics, chemistry and algebra ("About Us," n.d.)". Because of these altruistic goals, big IT giants like Microsoft, Google, and Facebook as well as the U.S. government supported it in terms of donations and participation.

Its eponymous website includes a set of puzzles that allows the students to apply programming concepts in a visual way with the use of blocks. It also includes teacher guides and curriculum plans that can be used in teaching different grade levels. Code.org mainly utilizes Blockly as its programming interface. Hence, Blockly and Code.org can be treated as one and in some pedagogical scenarios such as this study. They were launched together with the Hour of Code as an initiative of the U.S. government.

CodeCademy

CodeCademy is an education company with an eponymous website that also serves as an online interactive platform. Founded by Zach Sims and Ryan Bubinski in 2011, its mission is to build education that the world needs, which is a net native education ("About," n.d). It currently offers free interactive online classes for programming languages like Javascript, PHP, Python, and Ruby as well as mark-up languages like CSS and HTML.

Numerous technology websites and blogs have given good reviews for CodeCademy, which claims to be capable of turning anyone into a programmer (Segall, 2011). It also joined the Computer Science Education Week's Hour of Code in December 2013 by launching its free iPhone app that allows users to learn programming on their iPhone units (Summers, 2013).

The flowchart in Figure 1 shows the sequence of activities undertaken for the study using the framework presented by Teplechuk (2013). This framework divides the activities into Theoretical, Empirical, and Analytical categories. The details of the figure below are discussed in the Course Design and Data Analysis sections of this document.

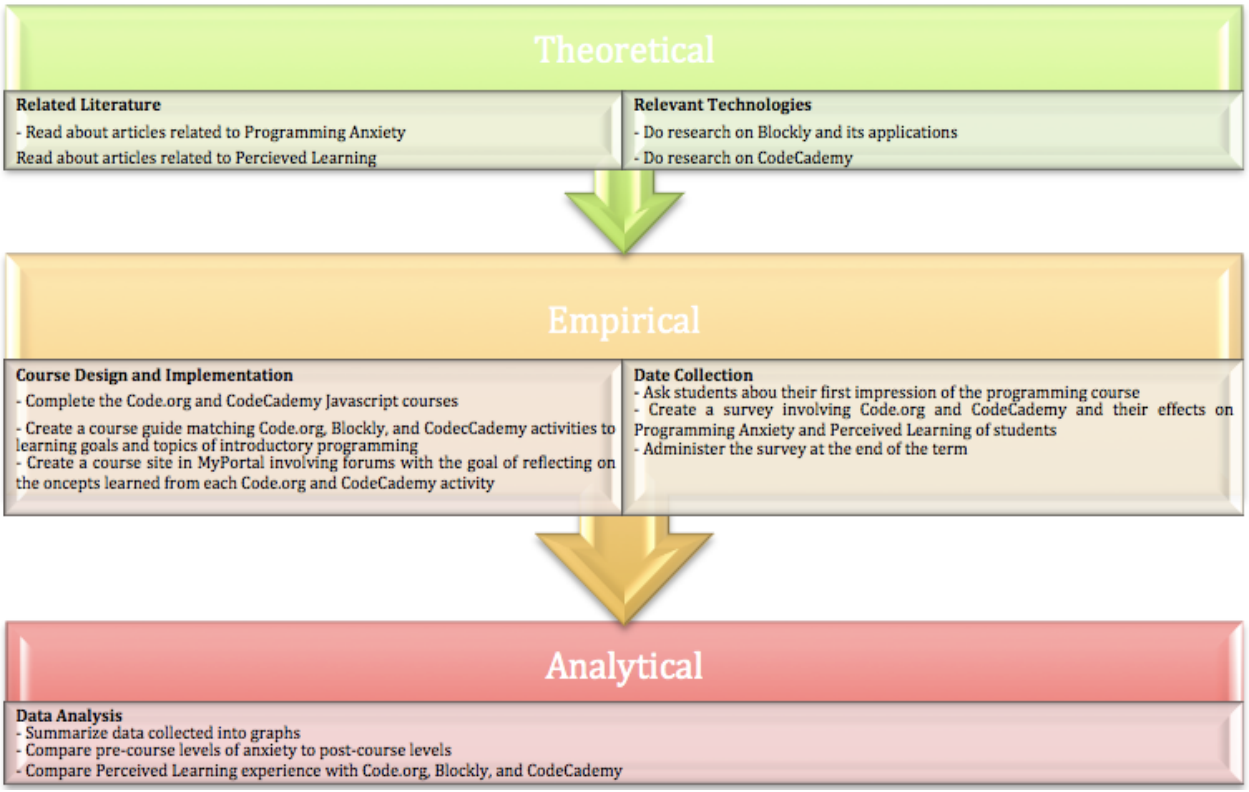

Figure 1. Methodology flowchart
Source: Patterned after Teplechuk (2013)

The Course Design

Before the course was offered, a review of literature pertaining to programming anxiety and interest in programming was done. From this review, a list of platforms was identified and eventually narrowed down based on their ability to address Jenkins' factors for programming anxiety, popularity, and accessibility for online and distance learning. Recommendations from experts were also considered as important factors for choosing the platforms. While Code.org and Blockly addresses the syntax before semantics and the choice of language issues explained by Jenkins in his discourse, CodeCademy offers problem-based activities, which let students acquire problem-solving skills while learning the syntax of the target programming language: Javascript. The videos in Code.org involving famous and respected celebrities from various fields having fun with learning how to program addresses the third issue stated by Jenkins, which is reputation and image.

The course design was crafted around these three platforms to ensure that the three modes of interactions according to Anderson and Garrison (1998) were highly achieved. Student-to-Content interaction was highly achieved because of the automated response and hints embedded in the online platforms' activities. Student-to-Student interaction was achieved through peer-evaluated forums where they showcase their programming creations and give advice to each other as to how to improve their work or solve a problem. It was also achieved through the group offline activities. Student-to-Teacher interaction was achieved through regular interviews and Q&A forums created to make the students reflect and synthesize what they have learned from the interactive activities provided by the three platforms.

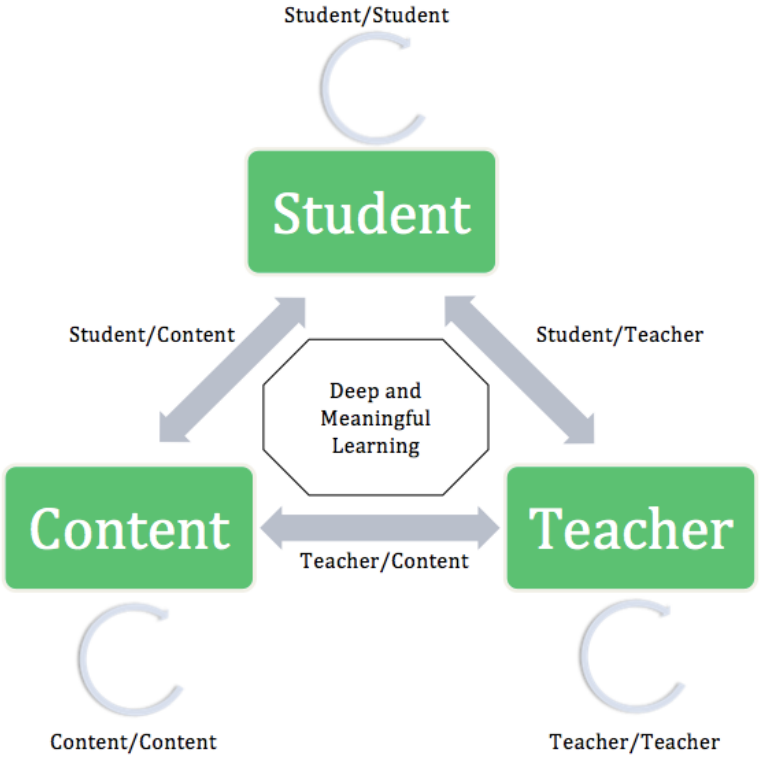

Figure 2. Modes of interaction in distance education from Anderson and Garrison (1998). Source: Image recreated from Anderson (2003)

The Course Implementation

During the start of the course, the students were asked questions related to their anxiety in learning programming. A free electronic textbook called Eloquent Javascript was prescribed to students as their main offline reference for the course.

Online materials using Blockly for teaching different concepts of programming such as variable, program sequence, branching, loops and functions were collected. Other concepts of programming such as algorithm design and abstraction were explained using Code.org's offline activities, which are open education resources (OERs). These concepts were discussed further through forums that were created with guide questions to better integrate the information derived from these activities. These Blockly-related activities were added as an introductory section to the existing materials and activities that are already being utilized to teach basic principles of programming. Supplementary activities were designed in Myportal, which is UPOU's learning environment, to ensure that the students were able to understand the concepts behind each stage in Code.org. Their progress was observed while they were taking the course using Code.org videos and activities. After a few weeks, they were asked again using an online questionnaire in the course site.

Random interviews were done in various consultation sessions in terms of how they were learning through CodeCademy, Code.org, Blockly, and the prescribed textbook. In each activity's forum, the side-comments and general feedback of the students were also collected and summarized.

After learning the basic concepts, the students were required to create their own profiles in CodeCademy. They were asked to complete the free course on Javascript, which reiterated the basic concepts taught in Code.org., but were also instructed to strictly use Javascript code instead of Blockly. Coding problems

explaining each concept were interactively presented to students as they complete each stage. In CodeCademy, structures like arrays and objects were also taught.

To test what they had learned, the students were required to submit a mini project that featured the basic principles taught in the course. This was done by asking them to make a web-based game using either traditional coding or Blockly.

Participants

The MMS 141 students enrolled during the 3rd Trimester of Academic Year 2013-2014 served as the respondents of this study. There were 52 students who took the course. They were all taking the Bachelor of Arts in Multimedia Studies (BAMS) program, under the Faculty of Information and Communication Studies. A total of 43 students participated in the first set of interviews and surveys. On the other hand, a total of 36 MMS 141 students comprised the respondents of the Programming Anxiety Survey, which was conducted towards the end of the course. Among them, there were 19 (52.78%) respondents who were female, while 17 (47.22%) were male. Additionally, 22 (61%) were within the bracket of ages 20-29 years old; six (16.68%) belonged to 30-39 age bracket, while eight (22.22%) were 40 years old and above.

Instruments

Several instruments were used to approximate the programming anxiety among the learners before, during, and after the course. The students were observed and their feedbacks were collected through interviews and forum interactions. These were used to supplement and validate the results of the main instrument, which was the Programming Anxiety Survey.

The survey contained 6 questions pertaining to learning anxiety, which were rephrased versions of some of the questions in the Computer Anxiety Scale. This widely-used scale for surveys conducted among computer-based learners was said to have an internal consistency (r=.95), which is considered to be high. According to Cohen and Waugh (1989), it was also negatively correlated with experience of people with computers. The validity of this new set of questions for programming anxiety has not yet been tested for validity, but was determined to be consistent with the results of the interviews among students before and after taking the course. Three of the questions contributed positively to programming anxiety, while the other three contributed negatively. The 6 questions and their respective contributions (+/-) were as follows:

1) I am anxious/nervous about learning programming. (+)
2) I am comfortable with participating in programming-related activities. (-)
3) I believe that learning programming is difficult. (+)
4) I welcome the possibility of having a job that involves programming in the future.(-)
5) I believe that I could/will not understand programming concepts well. (+)
6) I think that learning programming is enjoyable. (-)

Procedures

The Programming Anxiety Survey questions were asked for two sets: "Before Taking the Course" and "After taking the Course". For ease in discourse, "Before Taking the Course" will be identified as *set A*, while "After taking the Course" will be identified as *set B*. These sets were arranged in different sections of the same instrument to ensure dependence of the data sets. The students were asked to rate each statement using a 5-point Likert scale where 1 means that they strongly disagree and 5 means that they strongly agree.

Since the even numbered questions contribute negatively to the programming anxiety of the student, the formula *5-x+1*, where *x* is the score given by the student for the statement, was applied to them. After the transformation process, the average was computed from the scores given by each student for *set A* and *set B*. The differences of averages between the two sets were taken for each student and a Shapiro-Wilk test for normality was conducted and based on the result, the dataset was proven to observe a normal distribution. A paired t-test was then conducted to statistically test whether there was a change in programming anxiety levels after taking the course.

To further validate this, two questions were asked in the context of MMS 141 with its design of incorporating the interactive online coding platforms mentioned in this study. The first question was: "Was MMS 141 able to lessen your anxiety/nervousness/difficulty towards programming?" The second question was a follow up for those who answered "yes." It was stated as: "If so, how much?"

Finally, the students were asked to rank materials and tools according to their strength of impact on learning programming.

Results

From the preliminary interviews, it was revealed that 24 out of 43 of them were generally afraid of programming. After a few weeks of taking the course, 9 said that they became excited while 26 said that they became interested in learning programming. From summarizing their comments and interview replies, the following positive themes were formulated:

1) The format of the course was very encouraging and useful for them.
2) The interactive tools used in the course were very engaging.
3) The learners realized that they have the aptitude to learn programming.
4) The videos in Code.org were very encouraging.
5) The offline activities from Code.org allowed students to interact with one another and work together, which made the course more engaging and enriching.
6) The activities in CodeCademy, though more challenging, allowed them to transition from blocks to scripting/coding.

After analyzing the dataset coming from the Programming Anxiety Survey results, it was determined that the scores of set A had a mean of 3.19, and a standard deviation of 1.100850, while the scores of set B had a mean of 2.24, and a standard deviation of 0.753310. The mean of their differences was 0.95 while the standard deviation was 0.93.

After conducting paired t-test analysis on the Programming Anxiety Survey results, a t-value of 6.1493 and a p-value of 0 were obtained at 5% alpha and with 35 degrees of freedom. The null-hypothesis, which stated that there was no difference in mean programming anxiety ratings among students before they took the course compared to after they took the course, was rejected. Therefore, the observed difference was determined to be statistically significant.

Based on the results, it can be said that there is statistical difference between the average scores of students "*Before taking the course"* and *"After taking the course"*. Figure 3 below also shows that there was a significant decrease in average scores after taking the course, which also means a decrease in programming anxiety among the students.

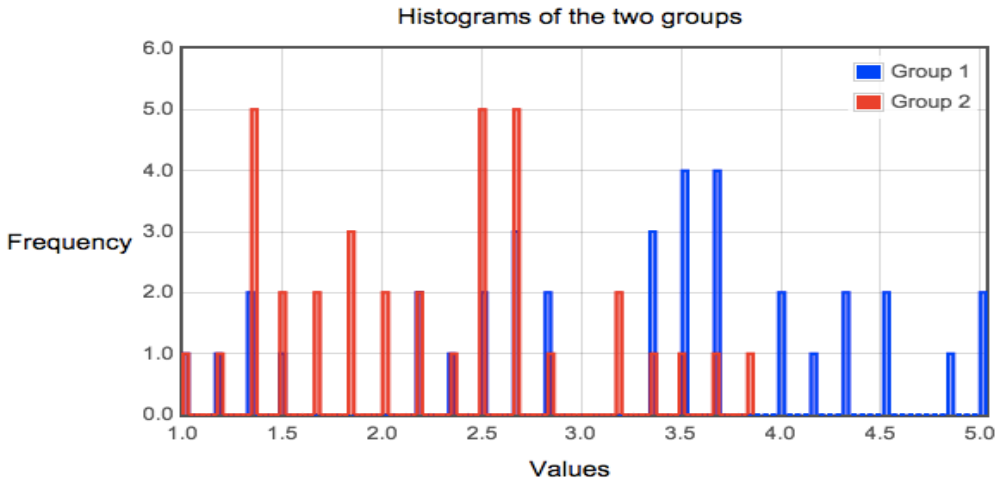

Figure 3. Histograms of "Before Taking MMS 141" set and "After Taking MMS 141" set.

The validation questions resulted in 32 out of 33 students saying that the course and its tools lessened their programming anxiety. More specifically, 13 said that their programming anxiety lessened by a large amount while 14 answered that their programming anxiety lessened by a fair amount. Finally, 5 of the respondents said that their programming anxiety lessened by a small amount. Additionally, they chose Blockly and Code.org as the materials to have the most positive impact in learning programming.

Discussion

The results support the claim of Neil Fraser who leads the team for maintaining Blockly in Google and who released Blockly on the Web in 2012. He said in an interview by Luis Ibanez (2015) that students face the syntax battle and the logic battle simultaneously when learning programming. Blockly eliminates the syntax battle by making it almost impossible for the students to commit syntax errors. This helps them to focus on winning the logic battle, which brings about a deeper understanding and appreciation of the core concepts of programming.

This can be explained by Jenkins' earlier notion that the linear and hierarchical approach of teaching syntax first before semantics (or logic) can cause anxiety and hence be a barrier to learning programming. What teaching with Blockly and Code.org did was to negate this issue by helping students develop programming logic and think computationally without having to wrestle with the difficulty of learning syntax.

It can also be said that the positive experience of students and their perceived learning when they used Blockly, Code.org, and CodeCademy in MMS 141 could be explained by the Interaction Equivalency Theorem introduced by Anderson (2003). The theorem states that a high level of at least one of the three forms of interaction (student-to-material, student-to-student, student-to-teacher) can bring about formal learning. Furthermore, eliminating or minimizing the other two forms will not degrade the learning experience. However, having more than one of these modes, despite being less cost or time effective, could provide a more satisfying experience for the learners. The interactive nature of the tools effectively heightened the level of interaction between the students and the material. Additionally, the student-to-student, and student-to-teacher interactions were also addressed with the forum and collaborative activities that were prescribed around these tools in the course design.

The automated feedback that CodeCademy provides the students whenever they make a mistake, the hints that it gives when they get into a dead-end, and the badges that it awards them for completing a certain amount of exercises amounts for a very high student-content interaction. This is also true with the interactive puzzles that Code.org provides.

Conclusion and Recommendation

Results have shown that there was a significant decrease in learning anxiety and an increase in perceived learning among the students who took the course.

The study has also shown that the use of online interactive platforms such as Blockly, Code.org, and CodeCademy, as well as the related activities in which they were utilized, effectively decreased programming anxiety among multimedia students of MMS 141.

Furthermore, Code.org, which was originally meant for K-12 students in the United States of America, was also observed as a suitable platform for teaching introductory programming in the Philippine university setting. It effectively made concepts less intimidating and provided a smooth transition towards the more cryptic aspects of the field.

MyPortal activities for reflection ensured that the students were able to critically process the concepts embodied in each Code.org activity. The videos from offline activities tapped into the creative talents of the students and made them regard programming as something that is not far removed from their field of multimedia studies.

However, comments from students suggested the use of more real-world applications of Blockly as some of them lacked confidence and found it challenging. It is therefore recommended that MIT App Inventor (http://appinventor.mit.edu/), an online android development tool that uses Blockly as its programming interface, be included as a parallel material in the next offering of the course. Improving the 6-question scale for Programming Anxiety to increase validity is also recommended. This could be done by including pre-test and post-test results in the next study.

References

Anderson, T. (2003). Getting the mix right again: An updated and rheoretical rationale for interaction. *The International Review of Research in Open and Distance Learning*, *4*(2). Retrieved from http://www.irrodl.org/index.php/irrodl/article/view/149/230.

Anderson, T., & Garrison, D.R. (1998). Learning in a networked world: New roles and responsibilities. In C. Gibson (Ed.), *Distance Learners in Higher Education* (p. 97-112). Madison, WI: Atwood Publishing.

About. CodeCademy. (n.d.). In *CodeCademy*. Retrieved from http://www.codecademy.com/about.

About Us. (n.d.). In *Code.org*. Retrieved from http://code.org/about.

Code.org Leaderboard (2015). In *Code.org*. Retrieved from http://code.org/leaderboard.

Brooks, K., & Lindgren, C. (2015). Responding to the coding crisis: From code year to computational literacy. In L.C. Lewis (Ed.), *Strategic discourse: The politics of (new) literacy crises*. Logan, UT:

Computers and Composition Digital Press/Utah State University Press. Retrieved from http://ccdigitalpress.org/strategic.

Cohen, B.A., & Waugh, G. W. (1989). Assessing computer anxiety. *Psychological Reports, 65*, 735-738.

Connolly, C., Murphy, E., & Moore, S. (2009). Programming anxiety amongst computing students - a key in the retention debate*? Education, IEEE Transactions*, *52*(1). Retrieved from http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4569871&isnumber=4774054

Delevett, P. (2014, March 7). Partovi twins quietly emerge as top Silicon Valley angel investors. *San Jose Mercury News*. Retrieved from http://www.mercurynews.com/business/ci_25297022/ali-hadi-partovi-twins-silicon-valley-angel-investors

Ibanez, L. (2015). Blockly makes it easier to learn to code. *Open Source [dot] com.* Retrieved from http://opensource.com/education/15/2/blockly-makes-easier-every-one-learn-code.

Jenkins, T. (2002). *On the difficulty of learning to program.* Retrieved from http://www.psy.gla.ac.uk/~steve/localed/jenkins.html.

Kim, B. (2015). Gamification in education and libraries. *American Library Association Techsource, 51*(2), 20-28.

Miller, C., & Wortham, J. (2011, March 26). In Silicon Valley, a Lack of Engineers. *NY Times*. Retrieved from http://www.nytimes.com/2011/03/26/technology/26recruit.html?_r=0

Owolabi, J., Olanipekun, P., & Iwerima, J. (2014). Mathematics ability and anxiety, computer and programming anxieties, age and gender as determinants of achievement in basic programming. *GSTF International Journal on Computing (JoC)*, *3*(4). Retrieved from http://dl6.globalstf.org/index.php/joc/article/view/381/401

Patterson, D. A. (2005). Restoring the popularity of computer science. *Communications of the ACM, 48*(9), 25-28.

Pinaroc, J. (April 4, 2008). Philippines Faces IT Skills Shortage. *Businessweek*. Retrieved from http://www.businessweek.com/stories/2008-04-04/philippines-faces-it-skills-shortagebusinessweek-business-news-stock-market-and-financial-advice

Rogerson, C., & Scott, E. (2010). The fear factor: How it affects students learning to program in a tertiary environment. *Journal of Information Technology Education*, *9*, 147-171. Retrieved from www.jite.org/documents/Vol9/JITEv9p147-171Rogerson803.pdf

Segall, L. (2011, November 29). Codecademy says it can turn anyone into a Web programmer. *CNN Money*. Retrieved from http://money.cnn.com/2011/11/29/smallbusiness/codecademy/index.htm

Summers, N. (2013, December 9). Codecademy: Hour of code iPhone app offers basic programming lessons. *TNW News*. Retrieved from http://thenextweb.com/apps/2013/12/09/codecademy-hour-code-app-iphone-teaches-basics programming-anytime-anywhere/

Teplechuk, E. (2013). *Emergent models of Massive Open Online Courses: An exploration of sustainable*

practices for MOOC institutions in the context of the launch of MOOCs at the University of Edinburgh [Unpublished MBA Dissertation]. University of Edinburgh, United Kingdom.