10-2013

Adaptive Gameplay for Programming Practice

Chris BOESCH
Singapore Management University, cboesch@smu.edu.sg

Sandra BOESCH
Pivotal Expert Pte Ltd, sandracboesch@gmail.com

Adaptive Gameplay for Programming Practice

Chris Boesch & Sandra Boesch
School of Information Systems (SIS) Singapore
Management University (SMU) Singapore
and Pivotal Expert, Pte. Ltd. Singapore

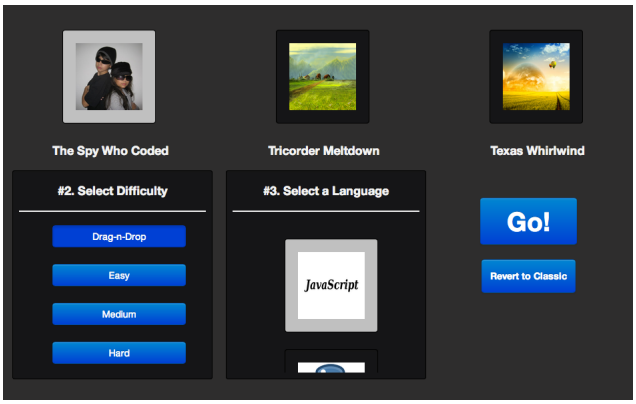
Abstract— **Over the past four years, we have collaborated to develop a set of online games to enable users to practice software languages in a self-directed manner and as part of a class. Recently we introduced a new adaptive difficulty feature that enables players to self-regulate the difficulty of the games they are playing to practice. These new features also provide additional information to further adapt the problem content to better meet the needs of the users.**

Keywords-education, programming, game-based learning

I. INTRODUCTION

When setting out to develop a more effective method to teach basic computer science, the authors were looking for innovative ways to provide additional, individualized feedback to students learning software languages such as Python, JavaScript, and Java for undergrad university courses. The authors took the approach to enable students to practice software languages on their own by having them solve short programming problems in an online game (see Figure 1) in a variety of software languages.

Figure1. SingPath Difficulty Selection Screen

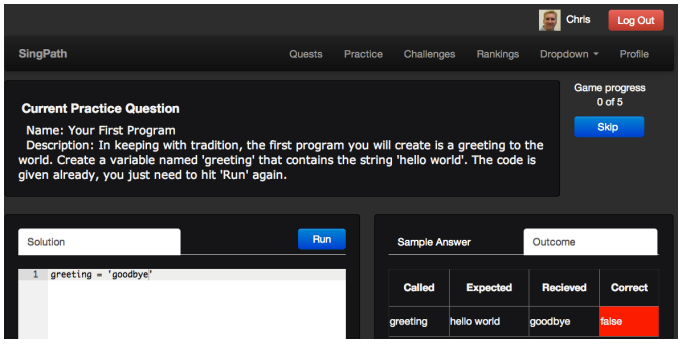


Students were able to practice solving these problems on their own time, from their own systems, wherever they had Internet access. This method enabled the authors to provide additional feedback to students in a more real-time manner than had been previously possible with live, in-class quizzes [1] and weekly problem sets turned in as homework. Students were still assigned problems to solve as in previous terms, but by requiring students to solve all problems in an online system, the authors were able to provide students with real-time feedback on their progress and at the same automatically track which students were on pace to solve all required problem prior to weekly deadlines.

As more people around the world began to solve problems on the online system, two consistent categories of feedback were received. A portion of the users continued to make comments such as "These problems are too difficult" while other users would provide comments such as "I am bored. Please let me skip the easy problems." To address this feedback, the authors created an adaptive difficulty mode. This enabled users to adjust the difficulty of problems to reduce the amount of boredom or frustration users might be encountering. The authors hypothesize that by enabling users to adjust the difficulty of the games they are playing, the users will be able to better balance their own boredom and frustration and increase the likelihood that they will be able to enter in to and stay in a state of learning flow [2] while practicing. Four difficulty modes were introduced: Easy, Medium, Hard, and Drag-n-drop. When users play on easy mode, they are provided step-by-step problems suitable for beginners. These problems often contain skeleton code to prompt the users and point them in the right direction for developing solutions. The hard setting presents the user with the hardest problems that have been loaded into the system for a given level. The relative difficulty of problems is determined by keeping track of how many attempts and how much time it takes all players to solve the problems. This enables the development of a ranking of relative problem difficulty. The medium difficulty level was designed to be adaptive to each user's individual skill level. This was accomplished by attempting to forecast how much time and how many attempts a user would require to solve problems in a given level. The problems that were considered to be easy or difficult were excluded. The remaining problems where considered medium difficulty problems. The initial settings for easy problems, based on past problem solving results, were set to be one or two attempts and less that sixty seconds. This meant that any problem that a user could solve with only one or two attempts within less than sixty seconds was considered an easy problem for the player. Similarly, hard problems were set to be any problems that required more than five attempts or more than five minutes to solve. This left the range for medium problems to be any problems that a user was able to solve in three to five attempts and in between one and five minutes. When there are insufficient problems for the user available that fall within the difficulty range the user is playing in, the remaining problems are selected from the next problem difficulty level(s). When playing in easy, if there are no

unsolved problems predicted to be easy for the player, the system selects from available medium problems to make up the difference. When the user is playing on the hard difficulty and insufficient problems are predicted to be difficult for the player, the remaining problems are selected from the problems that are predicted to be of medium difficulty for the user. And when the player is playing on the medium difficulty setting and insufficient problems are predicted to be of medium difficulty for the player, the problems to be solved are selected from a combination of the least easy and least hard problems forecasted for the user. Whenever insufficient easy, medium or hard problems are available for a player, the system makes a log entry indicating the need for additional content of a specific difficulty in a specific level. Over time, it is the goal of the researchers to use this data to guide the creation of additional problems at level of difficulty required by the majority of users.

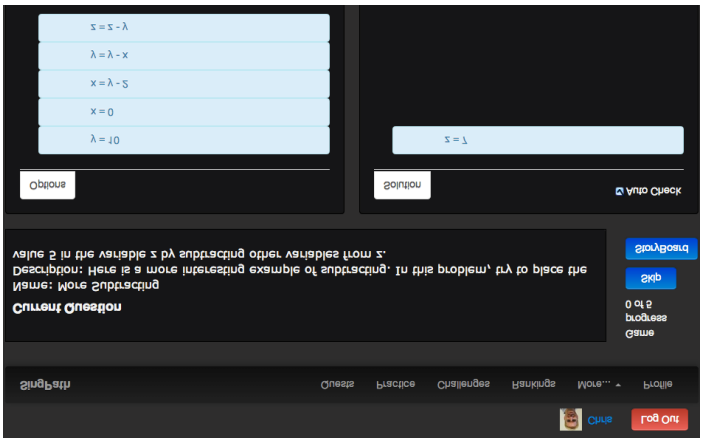
Figure 2. Easy Mode Problem Solving Screen



Even when the easy problems were created to be step-by-step problems with hints and guidance, some users still found the process of writing code for the first time to be difficult and stressful. The help alleviate this stress and encourage a wider audience to practice new software languages, the authors created a drag-n-drop mode that enabled users to assemble solutions from available lines of code rather than having to type them (see Figure 3). This had the dual benefits of creating an easier practice mode for beginners and a way to practice new software languages on mobile platforms such as tablets where keyboards might not be readily available and typing code could be a cumbersome experience. When users select to play a game in drag-n-drop mode, they are able to drag lines of code around and see feedback automatically rather than having to press a button to compile or test their solution. For users that have never programmed before, the drag-n-drop mode appears to be similar to puzzle games that users may have played on mobile devices in the past accept that in this situation the puzzles consist of lines of software code rather than colored jewels or other puzzle pieces. The authors hypothesize that after solving problems in drag-n-drop mode the users will be more comfortable reading software code in a new language when the move on to start writing their own code.

To encourage users to practice, the authors included a variety of gamification features such as badges, rankings, and completion metrics to complement the adaptive difficulty feature. They authors also added support for challenges, which require users to solve a specified number of problems at a specified difficulty level before unlocking a secret message. This challenge mode enables parents, mentors, and classroom facilitators to encourage users to practice with various learning outcomes in mind. Challenging users to practice in drag-n-drop mode can be used to raise the awareness of a programming language. Challenging users to play on medium difficulty will require users to play in a difficulty mode that adapts to their own individual capability.

Figure 3. Drag-n-Drop Difficulty



The ability to provide challenges to students could be used to support Team-based learning methods [3] where students are required to learn material on their own before coming together as groups in class to attempt solving more advanced problems.

II. Findings & Results

As the authors continue to collect more data and content, they hope to find ways to more consistently and automatically enable users to enter into a state of flow while practicing. While the ability for users to self-regulate is a first step, the medium mode of play is intended to automate the process of keeping users in a certain zone of challenge. However, it is not known if the current, arbitrary challenge zone of three to five attempts in one to five minutes is optimal for achieving flow. It is also unknown if there will be and optimal flow zone for a majority of users or if every user's flow zone will be different. The authors how to explore this design space in future research.

III. Discussion & Conclusions

In previous work, one of the authors found that highly-motivated students were able to achieve significant learning outcomes on their own [4] with minimal instruction. By providing a less boring and/or frustrating experience, it should be possible for users with less extrinsic motivation to practice longer and achieve higher levels of mastery. The introduction of drag-n-drop should also enable greater usage on mobile devices and enable the online game to be used in lab settings where only tablets are available. And as more users play, the data collected on the relative difficulty of different problems and the experience of users should enable the online game to be further enhanced.

ACKNOWLEDGMENTS

We are grateful to the Singapore Management University School of Information Systems for enabling us to conduct tournaments in live classroom settings. We are also grateful to the staff of Pivotal Expert for maintaining and enhancing the SingPath platform and for making it free to students and faculty around the world.

REFERENCES

[1] Boesch, Chris; Boesch, Sandra. Tournament-based Teaching. 4th Annual International Conference on Computer Science Education: Innovation and Technology (CSEIT 2012).

[2] Csikszentmihalyi, M. *Finding flow: The psychology of engagement with everyday life.* 1st edition. Basic Books: New York, NY;1997.

[3] Michaelsen LK, Knight AB, Fink LD. Team-Based Learning, A Transformative Use of Small Groups in College Teaching. 1st edition. Sterling, Virginia: Stylus Publishing, LLC; 2002.

[4] Boesch, Chris; Steppe, Kevin. *Case Study on Using a Programming Practice Tool for Evaluating University Applicants.* 3rd Annual International Conference on Computer Science Education: Innovation and Technology (CSEIT 2011).