Educational games for CS1: raised questions

Jamie McKee-Scott
Computer Science, UBC Okanagan
3333 University Way
Kelowna, BC, V1V 1V7

jamieleemckee@gmail.com

Lasserre Patricia
Computer Science, UBC Okanagan
3333 University Way
Kelowna, BC, V1V 1V7
(00) 1 250 807 9502

patricia.lasserre@ubc.ca

ABSTRACT

Although first year computer science courses commonly have relatively large class sizes and a high level of initial attendance, they very quickly succumb to large drop-out rates [1]. Following a similar model to the popular game *Brain Age*, we previously suggested a game system that could help students practice their programming skills every day [2]. Two simple games were implemented in the summer 2010.

The first game focuses on basic syntax for Java such as reserved words, and name conventions: the correct words are randomly displayed on the screen and mixed with words that have no particular meaning. The student must click only on the type of words requested (e.g. reserved words) while a timer limits the amount of time they have. To be successful and access the next level, a specific score must be reached. While the game is very simple, the question on how to maximize students' learning through the game is not. How much time should we give a student with no previous programming experience to discover all the correct words? What strategy can be used from one level to the next to help students learn? For example, is it a good idea on the first level to slowly remove all the words that are not reserved words so that they can learn the ones that are? Also, how should the next levels evolve? Should each level be specific to one topic only (reserved words, class naming, method naming), or should we build on the previous level such that the second level would not only include the class naming convention but also reserved words? And if so, what question should we ask the students? To identify only the possible classes or identify both reserved words and classes? To bring interest to the game in the fourth and last level the words move on the screen. But at which speed should the words move? And what is the value of the moving in terms of learning and entertainment?

The objective of the second game is to ensure students learn where a programming statement starts and ends, should it be a basic instruction, a conditional statement, loop, method or class. The game gives the student some Java code and asks him/her to identify the starts or ends of some of its elements in a specific amount of time. Again, how can we make the game progress in an effective way? While it is obvious that the various levels can evolve with the type of starts and ends to identify (e.g. loop vs. constructor), it is less clear how to ensure a student can learn it from the game. One option is to make it obvious that there are several parts in the programming statement by showing each individual part as a visible button (e.g. have two separate buttons, one for the statement and one for the semi-colon). The next level would then consist in hiding the buttons locations, eliminating the students' ability to blindly select. However, it is too obvious? Also the timing of each game interface must be evaluated.

A third game is planned for this summer that will allow the students to demonstrate their programming knowledge by asking them to order a series of elementary programming tasks. The student is provided with unordered lines of code, some of which may not be required to complete the task asked. Various levels of difficulty can be attained by providing or removing color guidance and method headers or by eliminating the useless lines of codes. But what is really effective in terms of learning?

The creation of those three simple games has raised important questions regarding how to attain the learning objectives of the game. To respond to those issues, a usability study will be conducted this summer. We hope to test each option with grade 12 students (no programming experience) and previous first term programming students who have been unsuccessful in the course.

Categories and Subject Descriptors

K.3.2 [**Computers and Education**]: Computer and Information Science Education – *Computer science education.*
K.8.0 [**Personal Computing**]: General – *Games.*

General Terms

Experimentation.

Keywords

Educational games, CS1

1. REFERENCES

[1] Kinnunen, P. and Malmi, L. 2006. *Why students drop out CS1 course?*. In Proceedings of the Second international Workshop on Computing Education Research (Canterbury, United Kingdom, September 09 - 10, 2006). ICER '06. ACM, New York, NY, 97-108. DOI= http://doi.acm.org/10.1145/1151588.1151604

[2] Patricia Lasserre and Kyle Kotowick. 2010. Proposal for a new strategy to practice programming. In *Proceedings of the 15th Western Canadian Conference on Computing Education* (WCCCE '10). ACM, New York, NY, USA, , Article 9 , 3 pages. DOI=10.1145/1806512.1806526 http://doi.acm.org/10.1145/1806512.1806526