Supporting the learning of programming in a social context with multi-player micro-games

Madeline Alsmeyer, Judith Good, Katherine Howland,
Graham McAllister, Pablo Romero and Phil Watten
Department of Informatics, University of Sussex

1 Introduction

As digital technology becomes indispensable and pervasive the need for software developers, both professionals and end-users, becomes greater and greater. However, admissions and retention rates to university computing courses are falling [9] and although end-user programmers are a thriving group there are serious concerns about the quality of the software which they create [2]. The former issue is particularly serious in the UK, where not only university but computing A level numbers are falling, the IT labour market is growing and as a result there is a predicted shortage of quality IT skills over the next few years [5].

An important part of the problem at introductory levels has to do with the perceived difficulty of mastering programming, a fundamental subject for software developers. One of the main problems is that there is a set of basic skills (like the meaning and use of variables, assignments, data types, etc.) that needs to be understood and applied at once from very early in the learning process. Novices very often have difficulties in understanding, integrating and applying these pieces of knowledge when trying to perform programming tasks [8, 7, 1].

An additional problem, which in some ways is more difficult to tackle, is the lack of a social context for the learning of programming. Research suggests that supporting social interaction is a powerful way to lower the barriers to learning programming [6]. However, it is often unclear to both the programming teacher and students how best to incorporate appropriate social learning into a subject that is most commonly evaluated through solo activity.

There are a number of approaches to making the introduction to programming easier and more appealing, and a remarkably promising one from these is using computer games. The common trend with this approach has been to provide environments and activities for novice programmers to create their own games [4, 6]. A less explored route has been developing games which require students to acquire programming skills in order to progress with the game-play.

2 Playing multi-player micro-games

Our hunch is that developing games of this sort is an effective way of introducing novices to programming as this is an approach that can help to tackle the two problems mentioned above. We aim to explore whether micro-games can help students to master basic programming skills, and support them in the integration of these skills to complete more complex tasks. Micro-games, short, self-contained video games around a specific topic, are increasingly included in commercial, popular game titles that claim to help train cognitive abilities with exercises for mental arithmetic, memory, logic, etc. [3] Micro-games can be employed to implement an incremental learning model in which students could start out practicing single, elementary programming skills. As students become more confident the micro-games can become more complex, requiring students to recognise and apply multiple skills. Micro-games could target generic rather than language specific programming skills. In this way they would be useful for students learning a

range of programming languages.

Additionally, micro-games are frequently implemented in portable platforms that enable casual, multi-player, real-time competition and cooperation. These features can be employed to help provide social interaction when learning programming. We are particularly interested in handheld game consoles (or equivalent platforms such as modern mobile phones or portable media players) as they can enable casual gaming in which different types of group interaction (additional to competition) can be enforced.

We aim to explore the effectiveness of a casual, multi-player, micro-games approach to foster skill development within an environment that supports social interaction. There are a number of research questions associated with this approach:

- What is the relationship between different types of group interaction that can be enforced by the platform and the effectiveness of the approach (in learning gains, attitudes, etc.)?
- Are there differences in the emotional and motivational experiences of learning a programming language through a traditional approach and learning programming with the addition of multi-player micro-games?
- Does the use of multi-player micro-game approach change students' perceptions of their potential abilities in programming?
- Does this platform of teaching and informal learning change the profile of students interested in studying programming?
- When, how and with whom do students choose to collaborate on program gaming tasks with other students?

We aim to address these questions in an empirical way by designing, implementing and evaluating multi-player micro-games built for portable gaming platforms.

References

[1] C. Bladek and F. P. Deek. Understanding novice programmers difficulties as a requirement to specifying effective learning environments. In R. Nata, editor, *New directions in higher education*, pages 1–22. Nova Science Publishing, 2005.

[2] M. Burnett, C. Cook, and G. Rothermel. End-user software engineering. *Communications of the ACM*, 47(9):53–58, 2004.

[3] I. Fuyuno. Brain craze. *Nature*, 447:18–20, 2007.

[4] M. Guzdial and E. Soloway. Teaching the nintendo generation to program. *Communications of the ACM*, 45(4):17–21, 2002.

[5] M. Holliss and M. Cadby. A study on the IT labour market in the UK. Accessed 27 June, 2008 from http://www.cphc.ac.uk/docs/reports/cphc-itlabourmarket.pdf, 2008.

[6] C. Kelleher and R. Pausch. Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers. *ACM Computing Surveys*, 37(2):83–137, 2005.

[7] A. Robins, J. Rountree, and N. Rountree. Learning and teaching programming: A review and discussion. *Computer Science Education*, 13(2):137–172, 2003.

[8] J. C. Spohrer and E. Soloway. Novice mistakes: are the folk wisdoms correct? *Communications of the ACM*, 29(7):624–632, 1986.

[9] J. Vegso. Interest in CS as a major drops among incoming freshmen. *Computing Research News*, 17(1):17–18, 2005.