Chapter 4
Principles of a Casual Serious Game to Support Introductory Programming Learning in Higher Education

Adilson Vahldick
Universidade do Estado de Santa Catarina (UDESC), Brazil

Maria J. Marcelino
Universidade de Coimbra, Portugal

António J. Mendes
Universidade de Coimbra, Portugal

ABSTRACT

Casual games are characterized for their fast learning curve. Casual games tasks usually are short and have increasing difficulty. This seems an interesting approach to learn and practice introductory computer programming concepts for students that face difficulties. Many of serious games intended to support computer programming learning are commercial and aimed at children. Also only a few of those described in the literature are available to teachers. This chapter describes the development of a new game that aims to support introductory computer programming learning and its pilot study with three undergraduate introductory classes. The chapter proposes a set of design principles that might be useful in the development of casual games to support computer programming learning. These principles resulted from the experiment and include game features that were considered important to engage students and to improve some students' computer programming skills.

INTRODUCTION

Worldwide demand for Computer Science (CS) professionals is growing, and despite Information Technology (IT) is so embedded in people's everyday lives, it is observed that the demand for computer and information sciences degrees does not follow it (Litan, 2015; Looksharp, 2015). Students figure out early that learning computer programming is hard and a very complex subject (Gomes & Mendes,

DOI: 10.4018/978-1-5225-1034-5.ch004

2007). Governments have introduced initiatives to teach concepts of Computational Thinking (CT) from an early age in schools as a way to meet the demand (Code.org, 2015). However, what about the current undergraduate students? For a long time the problems and difficulties in introductory programming learning have been researched, and some solutions have been proposed (more appropriate languages, improved pedagogic strategies, visualization and simulations tools, among others) (Du Boulay, 1986; Winslow, 1996; Mannila et al., 2006; Pears et al., 2007; Shaw, 2012). Anyway, some studies contend that the key factor in promoting programming learning is persistency and the student intensive and disciplined practices (Robins et al., 2003; Lahtinen et al., 2005). This requires that the student has a high level of motivation, which must be stimulated by the teacher and the pedagogic environment. Individualized scaffolding by teachers, either in person or by intelligent environments that replicate their presence (Piech et al., 2012), rich pedagogical contexts, for example focusing the programming exercises in areas that interest the students (Gomes & Mendes, 2015). Some undergraduate courses have adopted the development of small games as motivators (Bayliss & Strout, 2006; Barnes et al., 2007; Morrison & Preston, 2009). Other introduced the use of games to reinforce concepts, to support the development of abstract concepts, bringing theory and practice closer, and as an alternative to the traditional programming exercises (Eagle & Barnes, 2009; Muratet et al., 2010; Lee et al., 2014).

Malliarakis et al. (2014) and Vahldick et al. (2014) made reviews on serious games for computer programming learning. They identified the programming topics covered by these games, learning activities, technologies and grouped them by type: LOGO-like games (action games where the player programs the movements of a robot, a turtle or other kind of character in a simulated world), Adventure games (player commands a hero to explore the world, to collect objects and to interact with other characters controlled by the game) and General Puzzles' games (the remainder of the analysed games belong to several types, such as simulations, real-time strategies (RTS) and maze games). There is another aspect as important as those cited above, which is the game's learning curve, meaning the time students need to learn the game features and mechanics instead of learning the material (Landers & Callan, 2011). Casual Games (CG) are easy to play and require few instructions (Kadle, 2009). These games are developed following design principles and heuristics to smooth their learning curve (Juul, 2010; Trefry, 2010). On the other hand, Hardcore Games (HG) are those in which the player is inserted in a virtual environment (typically 3D) wherein she/he spends a longer time in learning the rules, mechanics and tools of the game (Wang & Sun, 2011). Another important difference is the session time: while in CG the player finishes an assignment within minutes, the tasks to be completed in a HG are designed to involve more requirements, with increased run sequence steps (sub-goals) and therefore a larger amount of time is necessary (Landers & Callan, 2011; Chiapello, 2013). Both types of games are worth as serious games: HG to learn procedural knowledge, or as a full scale learning game to cover a substantial portion of a course or to train a company's department; CG as a complementary tool to learn conceptual or factual knowledge (Kadle, 2009; Smith & Sanchez, 2010).

A number of studies propose guidelines, design and assessment principles for serious games (Bayliss & Schwartz, 2009; Linehan et al., 2011; Bellotti et al., 2013). Some are focused on computer programming learning (Bayliss & Schwartz, 2009; Kazimoglu et al., 2013; Lee et al., 2014). These guidelines describe how to design game elements (mechanics, rules, and aesthetics) with instructional concerns (feedback, learning theories, learning tasks, scaffold) and curriculum settings (institutional, classroom environment). This chapter introduces a concrete set of casual games' design principles for computer programming learning, linking instructional issues with game elements. These principles are the result