The Appropriateness of Scratch and App Inventor as Educational Environments for Teaching Introductory Programming in Primary and Secondary Education

Stamatos Papadakis, Secondary Education Teacher, Crete Greece
Michail Kalogiannakis, Department of Preschool Education, Faculty of Education, University of Crete, Crete Greece
Vasileios Orfanakis, Secondary Education Teacher, Crete Greece
Nicholas Zaranis, Department of Preschool Education, Faculty of Education, University of Crete, Crete Greece

ABSTRACT

Teaching programming is a complex task. The task is even more challenging for introductory modules. There is an ongoing debate in the teaching community over the best approach to teaching introductory programming. Visual block-based programming environments allow school students to create their own programs in ways that are more accessible than in textual programming environments. These environments designed for education allow students to program without the obstacle of syntax errors (errors in typing commands) found in traditional text-based languages. In this paper, the authors focus on the use of App Inventor and Scratch as blocks-based programming environments designed explicitly with novices in mind. In the authors’ analysis, both Novice Programming Environments (NPEs) seemed to be attractive platforms for introducing fundamental concepts in computer programming and both look appealing for both majors and non-majors.

KEYWORDS

App Inventor for Android (AIA), Novice Programmers, Novice Programming Environments (NPEs), Primary Education, Scratch, Secondary Education

INTRODUCTION

The power of computers stems from their ability to represent our physical reality as a virtual world and their capacity to follow instructions with which to manipulate that world (K–12 Computer Science Framework, 2016). Ideas, images, and information can be translated into bits of data and processed by computers to create apps, animations etc. Games and apps that teach programming skills are even available on smartphones (Papadakis, Kalogiannakis & Zaranis, 2017a).

In recent years, Computational Thinking (CT) and related concepts (e.g. coding, programming, algorithmic thinking) have been promoted by educational stakeholders as skills that are as fundamental for all as numeracy and literacy (Benton, Hoyles, Kalas, & Noss, 2017; Bocconi, Chiocciariello, Dettori, Ferrari, & Engelhardt, 2016; Papadakis & Kalogiannakis, 2008; Papadakis, Kalogiannakis, Zaranis, & Orfanakis, 2016). A number of prominent institutions and industry inside and outside Europe (Papadakis, 2016) have intervened in the debate about the introduction of CT skills in compulsory education (Royal Society, Académie des Sciences, Informatics Europe and the Association of

DOI: 10.4018/IJWLTT.2017100106
Computing Machinery (ACM), Working Group on Informatics Education) (Bocconi et al., 2016). Additionally, computing and mathematical occupations are projected to be among the fastest-growing of any occupational group, with 1.3 million job openings in computing-related fields projected by 2022 (Richards, & Terkanian, 2013).

Although, in general, it is agreed that CT and programming are not overlapping sets (Wing, 2006) programming can make CT concepts concrete and become a tool for learning. Despite this widespread interest in developing CT skills among schoolchildren due to the fact that the role and significance of computing has increased in society and the economy (Kalogiannakis, 2010; Wilson, Sudol, Stephenson, & Stehlik, 2010; Papadakis, Kalogiannakis, & Zaranis, 2016b) student motivation to enrol in computing fields is in decline (Karakuş, Uludag, Guler, Turner, & Ugur, 2012). A reason is that novices face many barriers when learning to program a computer, including the need to learn both a new syntax and a model of computation (Hundhausen, Farley, & Brown, 2009; Xinogalos, Satratzemi, & Malliarakis, 2017). The language syntax, the seemingly esoteric punctuation and formatting rules that must be followed when composing programs, can be a serious barrier for novice programmers (Weintrop & Wilensky, 2015).

As Guzdial (2004, p. 152) states the target in introductory courses is toward making programming more interesting, more relevant, and more powerful for students-novices. Different kinds of activities – game creation, robot programming, and storytelling - have been proposed for different educational levels, obviously with different complexity (Bocconi et al., 2016; Papadakis & Orfanakis, 2017). In an attempt to increase interest in computer science (CS), much effort has been made developing tools, activities and preliminary learning materials, both for younger children (Zaranis, Kalogiannakis & Papadakis, 2013; Papadakis, Kalogiannakis & Zaranis, 2016a; 2017b) and for high schools and universities (Meerbaum-Salant, Armoni, & Ben-Ari, 2013). Research has shown that NPEs (Novice Programming Environments) or ILEs (Initial Learning Environments) play an important role in attracting and retaining new programmers’ interest in computing as they manage to engage students and improve their attitudes towards CS (Papadakis, Kalogiannakis, Orfanakis, & Zaranis, 2014; Orfanakis & Papadakis, 2014; 2016).

A reason is that the use of program visualization with educational software tools such as Scratch and App Inventor is gaining acceptance as an educational approach, particularly in K-12 classrooms (Tabet, Gedawy, Alshikhabobakr, & Razak, 2016). Visual block-based programming environments allow school students to create their own programs in ways that are more accessible than in textual programming environments (Dwyer, Hill, Hansen, Iveland, Franklin, & Harlow, 2015). Additionally, although these environments have been designed for both young students and beginning programmers, they also allow students to build complex programs, games, apps, and animations (K–12 Computer Science Framework, 2016).

The remainder of this article is organized as follows. After reviewing related work in Section Background, we present the two different NPEs in Section What is Scratch? and What is App Inventor? A summary of the similarities and differences between the two NPEs is presented in the following section. Conclusions and discussions are given in the last section.

BACKGROUND

As computing has become an integral part of our world, public demand for computer science education is high (K–12 Computer Science Framework, 2016). Numerous papers examine the potential advantages of introducing CT in compulsory education (Papadakis et al., 2014). The belief is that CT can enable children and young people to think in a different way while solving problems,
Emergent Knowledge Artifacts for Supporting Trialogical E-Learning
www.igi-global.com/article/emergent-knowledge-artifacts-supporting-trialogical/2986?camid=4v1a

Enhancing Teaching and Learning Computer Hardware Fundamentals Using PIC-Based Projects
www.igi-global.com/chapter/enhancing-teaching-learning-computer-hardware/30431?camid=4v1a

Facilitating Learning by Going Online: Modernising Islamic Teaching and Learning in Indonesia
www.igi-global.com/chapter/facilitating-learning-going-online/74973?camid=4v1a