Available online at www.sciencedirect.com

SCIENCE DIRECT®

Computers & Education 47 (2006) 148–171

COMPUTERS & EDUCATION

www.elsevier.com/locate/compedu

An introduction to object-oriented programming with a didactic microworld: *objectKarel*

Stelios Xinogalos [a], Maya Satratzemi [a,*], Vassilios Dagdilelis [b]

[a] *Department of Applied Informatics, University of Macedonia, 156, Egnatia str., P.O. Box 1591, 54006 Thessaloniki, Greece*

[b] *Department of Educational and Social Policy, University of Macedonia, 156, Egnatia str., P.O. Box 1591, 54006 Thessaloniki, Greece*

Received 27 May 2004; accepted 15 September 2004

Abstract

The objects-first strategy to teaching programming has prevailed over the imperative-first and functional-first strategies during the last decade. However, the objects-first strategy has created added difficulties to both the teaching and learning of programming. In an attempt to confront these difficulties and support the objects-first strategy we developed a novel programming environment, *objectKarel*, which uses the language Karel++. The design of *objectKarel* was based on the results of the extended research that has been carried out about novice programmers. What differentiates it from analogous environments is the fact that it combines features that have been used solely in them: incorporated e-lessons and hands-on activities; an easy to use structure editor for developing/editing programs; program animation; explanatory visualization; highly informative and friendly error messages; recordability. In this paper, we present the didactic rationale that dictated the design of *objectKarel* and the features of the environment, including the e-lessons. In addition, we present the results from the use of *objectKarel* in the classroom and the results of the students' assessment of the environment.

Keywords: Programming and programming languages; Teaching/learning strategies; Pedagogical issues

[*] Corresponding author.
E-mail addresses: stelios@uom.gr (S. Xinogalos), maya@uom.gr (M. Satratzemi), dagdil@uom.gr (V. Dagdilelis).

0360-1315/$ - see front matter © 2004 Published by Elsevier Ltd.
doi:10.1016/j.compedu.2004.09.005

1. Introduction

The three basic strategies for an initial approach to teach programming are: imperative-first, functional-first, and objects-first. The first two strategies have been used for a fairly long period of time, whereas the third one appears to have attracted interest in the last few years. The functional-first strategy initially places emphasis on functions leaving the presentation of state for later, whereas in the imperative-first strategy the emphasis is first given to the state and then the concept of functions is presented.

As far as the objects-first strategy is concerned, according to the ACM curricula report, "Objects-first emphasizes the principles of object-oriented programming and design from the very beginning. The objects-first strategy begins immediately with the notion of objects and inheritance and then goes on to introduce more traditional structures" (Chang et al., 2001, p. 30). This means that from the very beginning both the state and functions must be presented.

As the authors of the ACM curricula report acknowledge, the objects-first strategy creates added difficulties to both the teaching and learning of programming. The classic methodology for the teaching of programming began with small and simple programs to be followed by more complex and larger-sized ones. This approach gave novice programmers time to assimilate and to gradually build up new knowledge relevant to the development of programs. However, when using the objects-first strategy, students are required to work with objects from the very beginning. This means that from the very beginning they will have to be taught about objects, classes, methods, constructors, inheritance and at the same time they will have to be taught the concepts of types, variables, values, as well as having to learn the syntax of the language which, as has been shown in the research, comprises one of the biggest sources of difficulties for novice programmers.

In an attempt to support the objects-first strategy various educational software tools have been developed, such as: BlueJ (Kolling, Quig, Patterson, & Rosenberg, 2003), Karel J. Robot (Bergin, Stehlik, Roberts, Pattis, & Karel, 2004), Jeroo (Sanders & Dorn, 2003), JKarelRobot (Buck & Stucki, 2000), and Alice (Cooper, Dann, & Pausch, 2000). Certain of these tools, like Karel J. Robot, Jeroo, JKarelRobot, and Alice, constitute programming microworlds which are based on a physical metaphor, while, BlueJ is an integrated programming environment whose main feature is that the user begins with a set of predetermined classes and can create objects and call on methods for those objects in order to examine their behavior.

In our attempt to support the objects-first approach we developed a novel programming environment *objectKarel* (Xinogalos, 2002), which uses the language Karel++, as defined by Bergin, Stehlik, Roberts, and Pattis (1997).

ObjectKarel consists of a programming microworld and thus belongs to the first category mentioned above. However, *objectKarel* incorporates certain characteristics, which do not exist in other software of the same category but which facilitate both the teaching and learning of OOP. The two basic characteristic components of *objectKarel* are the following:

- It is a programming environment, which helps the student to develop programs easily. Program development is accomplished with the help of a structure editor where by selecting the appropriate commands from menus the development/editing of a program takes place.