A Multiple Representational Environment for Learning Programming and C

Konstandina Zikouli[1] , Maria Kordaki[1,2] and Elias Houstis[3]
[1] Dept. of Computer Engineering & Informatics Univ. of Patras, 26500, Patras, Greece
[2] Computer Technology Institute, 26500, Patras, Greece,
[3]Dept. of Computer Engineering & Telecommunications, Univ.of Thessaly, Volos, Greece
e-mail: zikouli@ceid.upatras.gr, kordaki@cti.gr, enh@inf.uth.gr

Abstract

In this paper we present the design and the basic features of a computer Learning Environment for programming and C using Geometrical Objects (L.E.C.G.O.). The design of this environment was the result of modeling. Basic aspects of constructivism have also been taken into account in its design. L.E.C.G.O. provides the pupils with opportunities to: a) express their problem solving strategies in multiple representation systems starting from intuitive representations and moving gradually to more sophisticated ones, b) solve a variety of familiar and meaningful problems, and c) overcome the cognitive load of the syntactical rules of programming in C by using appropriately designed computer-based authoring tools. A similar environment has not yet been reported.

Programming is not just a specific topic among others in the computer science and engineering (CS&E) curriculum but a 'mental tool' of general interest. Programming can be divided into four steps: a) comprehension of the problem at hand, b) defining a solution to that problem, initially in any form such as text-based, math-based, pseudocode, flow-chart, c) translation of that form into a selected programming language, and d) testing and debugging of the resulting program. Pupils have serious difficulties in performing all steps mentioned above. Good performance in programming implies the learners' ability to use different and new representation systems in order to express their problem solving strategies. The existing studies indicate that there is a need for a novice-oriented programming environment.

The general design of L.E.C.G.O. arose as a transformation of the theoretical considerations regarding three models: a) the *learning model* that reflects the designers' interpretations of constructivism and social views of learning, b) the *model of the subject matter* that refers to the basic aspects of programming and C, c) the *learner model* that describes the possible learners' actions and difficulties in learning programming and C. The construction of the models above emerged from the literature. Comparing the design of L.E.C.G.O. to other learning environments reported in the literature we stress: a) The existence of multiple representations that facilitate pupils to smoothly bridge the gap between intuitive and formal solutions b) Problem solving activities taken from the familiar and meaningful context of Geometry which minimizes the cognitive load of the student providing her/him with motivation, hands-on experience as well as with essential intrinsic feedback. The multiple representation systems provided by L.E.C.G.O. are presented below :

1)*Drawing-visual representations using hands-on experience.* Students can use all tools included in Cabri-Geometry II to express their intuitive knowledge regarding their solutions to the given geometrical tasks.

2)*Free-text based representations.* Here, pupils can express their solution strategies using the familiar symbolic system of natural language. This step calls pupils to reflect on their previously acquired hands-on experience.

3)*Text based representations using imperative and specific expressions*. Here, pupils have to transform their free-text based solutions in the imperative.

4)*Pseudo-code representations using basic algorithmic structures*. The previously constructed representations can be transformed in pseudo-code by using a set of buttons showing specific algorithmic structures.

5)*Representations in C.* A number of authoring tools showing the skeletons of basic components of a program in C are provided to help pupils in writing their programs.

6)*Graphical output of the written programs.* After the pupils finish writing the code they in the position to compile the program and see the visual results of their programming attempts. This feedback can help them to take control of their learning.

All representation systems above, except for the last one, were designed to act as 'transitional' representation systems to fill the gap between the pupils' concrete graphical solutions and the symbolic ones written in C. These 'transitional' systems were designed to act as scaffolding elements for those pupils who couldn't directly express their solving strategies in C. Pupils can express their own thinking by using the first and the second representation systems above while they can explore the thinking of others by using the remaining representations.

L.E.C.G.O. was implemented using Microsoft V.B. 6.0.