The role of multiple representation systems in the enhancement of the learner model in open learning computer environments

Maria Kordaki[1,2]

[1] Department of Computer Engineering and Informatics, University of Patras, 26500, Rion, Patras, Greece
[2] Research Academic Technology Institute, 26500, Rion, Patras, Greece

The construction of computer learning environments that support a variety of learning styles is crucial because learning is acknowledged as being a subjective activity strongly dependent on the learning style of each individual learner. Designers of computer learning environments emphasize the design of an appropriate learner model that can support learners in successfully expressing their inter-individual learning differences. This study demonstrates that the use of Multiple Representation Systems (MRS) can play a crucial role in the enhancement of the learner model so as to allow a variety of learners to express their knowledge regarding a specific learning subject. More specifically, this study presents: a) the learner model in solving problems using MRS integrated in an open learning computer environment constructed for the learning of programming and C by secondary level education students (The L.E.C.G.O. environment, [1]), and b) the learner model in solving the same problems in both the paper and pencil environment and a typical compiler of Turbo C.

Keywords: Multiple representation systems; Programming; Programming language C; Learners' model

1. 1ntroduction

Many researchers acknowledge that learners' individual differences are linked to their inequality in learning and success at school. More specifically, learners seem to arrive at schools with different learning styles, such as: intuitive, visual, holistic, field dependent, reflective, rational, analytic and field independent. Most learners' difficulties were attached to the gap between their intuitive knowledge and the knowledge they need to express in the proposed representation systems (RS) for use [2]. For example, prepositional, symbolic and abstract RS prevent some learners (usually beginners) from expressing their knowledge, the same systems being intended for use by advanced learners. Contrariwise, metaphors of everyday life and visual RS are more suitable for beginners. MRS are proposed for use to enable individual learner variety as well as to enable each learner to acquire a broader view of the subject to be learned [2-3]. In addition, multiple and linked RS provide the learner with the opportunity to study how the variation in one system can affect the other. In this way each learner can make connections between different aspects of a learning subject [2].

A number of computer learning environments provided with MRS are reported as successful in teaching and learning of a variety of subjects [4-5]. The success of these environments is mainly based on their ability to provide a variety of learners with learning opportunities, at the same time allowing them the opportunity to solve a variety of problems. Such environments can be characterized as Open Learning Computer Environments (OLCE). The design of OLCE is usually based on a modeling process [6-7]. More specifically, the designers have to construct three models, namely: 1) the model of learning, 2) the model of the subject to be learned, and 3) the learner model. The construction of the first model is based on modern theories of learning while at the same time interpreting them in terms of operational specifications. The construction of the second model emphasizes the essential concepts of the learning subject as they emerge from the literature. At this point, the selection of essential learning activities is crucial. The third model is constructed in such a way as to provide learners with opportunities to express their indi-

[1] Corresponding author e-mail: kordaki@cti.gr

vidual differences in performing the selected learning activities. At this point, the role of MRS in the enhancement of the learner model in performing essential learning activities for each specific subject is obvious; the selection of MRS is closely related to the subject to be learned. In this study, the enhancement of the learner model is approached by providing learners with MRS integrated in an OCLE. Such an approach for the enhancement of the learner model has not yet been reported. In particular, a number of RS are integrated in an OLCE, namely, the L.E.C.G.O. environment (Learning Environment for programming and C using Geometrical Objects, [1]). This environment was designed for the learning of basic aspects of programming and C by secondary level education pupils. In the context of L.E.C.G.O., programming is not viewed as being merely a specific topic among others in the computer science and engineering (CS&E) curriculum but as a 'mental tool' of general interest [8]. Along its interest, programming is a complex task including understanding, method finding and coding [9]. According to Winslow (1996) it can be divided into four steps: a) comprehension of the problem at hand, b) defining a solution of that problem, initially in any form, such as text-based, math-based, pseudocode, flow-chart, etc. c) translation of that form into a selected programming language, and d) testing and debugging of the resulting program [10]. Good performance on programming implies learner ability to use different and new RS in order to express their problem-solving strategies [9]. Learners are rarely aware of the problems that can be solved by a computer and of how they can benefit from using programming [11]. Learners also have serious difficulties in performing all the steps mentioned above [11-12]. Based on the above, L.E.C.G.O. was designed to allow learners to: a) express their problem-solving strategies in MRS, starting from intuitive representations and moving gradually to more sophisticated ones, b) focus on algorithmic solutions rather than syntactic rules of the programming language C, c) be engaged in problem-solving settings including a variety of familiar and meaningful problems, and d) correct their programming attempts by visualizing the program and its output [13-15].

The design methodology of L.E.C.G.O. was based on the construction of the three models described above: a) the learning model that was based on constructivist and social views of learning [16-17], b) the model of subject matter including an adequate subset of the programming language C and c) the learner model for problem solving using programming in C, while at the same time taking into account individual learner learning differences and difficulties [10], [12].

In the next section of the paper, the MRS included in L.E.C.G.O. are presented, followed by reference to the learner model in terms of solution strategies that could be constructed by students using the MRS provided. Next, the learner model in solving the same problems in both the paper and pencil environment and in a typical compiler of Turbo C is demonstrated. Subsequently, these models are discussed and conclusions are drawn.

2. MRS provided by the L.E.C.G.O. environment

L.E.C.G.O. consists of two main parts: a) the information part and b) the 'working space' part. In the information part, learners have the opportunity to access information about programming and C organized in four layers using MRS of information as follows: a) real-life holistic examples, b) examples selected for the learning of a specific concept, c) basic terms of programming and C in the form of a vocabulary, and d) sources of wider information eg. articles, and URLs. The four layers are hyper-linked.

The 'activity space' is organized in a such a way as to provide learners with opportunities to express their problem solutions in MRS. These systems can act as scaffolding systems, helping learners to smoothly pass from intuitive graphical solutions to more programming-oriented ones. The activities are selected from the context of drawing using geometrical objects. Learners can develop a strong motivation by drawing while using geometrical objects which are familiar to them. This context is also rich enough to pose gradually more complicated problems [1]. The RS provided are presented below:

1.G.R.S: *Graphical representations*. Here the tools of the well known *Cabri Geometry II* [7] environment are provided to the learners to form primitive, graphical solutions to the learning activities. By using these tools, learners can form a variety of both drawings using geometrical objects and typical geometrical constructions. The coordinates of these constructions can be automatically illustrated and used as input-

data in other RS. By using GRS, learners have the opportunity to express primary, intuitive solutions to the given activities through acquiring a kind of hands-on experience [18].
2. *F.T.R: Free text-based representations*. Here as well, learners can interpret their graphical solutions in their familiar system of natural language using a text editor. This activity asks students to reflect on their previous hands-on experience using GRS. The produced text is a 'transitional' representation that, like the previous graphic one, can act as a scaffolding element to help pupils gradually develop the ability to express their problem-solving strategies in the programming language C.
3. *ITR: Text-based representations using imperative and specific expressions.* Here, learners are asked to express their solutions in the imperative, also using a text-editor. This activity can help learners to move from a first person singular or plural situation to another in which the learner gives directions to be performed by the computer. Moreover, a number of specific expressions in the imperative are provided to help learners select those most appropriate for their solutions. These expressions are not immediately visible but appear by clicking on a button, whenever learners are unable to perform this step on their own.
4. *P.C.R: Pseudo-code based representations*. Here, learners are asked to express their imperative-based solutions in pseudo-code. An editor and a number of authoring tools in the form of buttons are provided. Each of these buttons shows a basic algorithmic structure or a geometrical figure on the corresponding text-box. By clicking on these buttons, pupils have the possibility of viewing the corresponding pseudo-code. It was decided to liberate the learners from the cognitive load of remembering syntactical rules.
5.P.L.C: Representations in the programming language C. Here as well, an editor and a number of authoring tools in the form of buttons showing the skeletons of basic components of a program in C e.g. '*basic structure of a program in C*', '*for*', '*while*' and '*if*" statements, as well as showing basic graphic commands e.g. '*line*', '*circle*', '*rectangle', 'ellipse'* and 'triangle', are offered. By clicking on these buttons, learners have the possibility of viewing the corresponding code. All these tools can help learners focus on the essential elements of the language, at the same time reducing the cognitive load arising from the need to recall specific syntactic rules and commands of the language.
6. *Graphical output of the written programs.* After the learners finish writing the code, they can save, compile and run their programs and see the results of their programming attempts on another window on the computer screen. Learners can then verify or correct their programming attempts by comparing this graphical output with their graphical solutions performed using GRS.
For the implementation of L.E.C.G.O, Microsoft Visual Basic 6.0 was used. The final program generated in L.E.C.G.O is compiled by the "Turbo C 2.01" compiler. Every action of the pupil is also recorded in a log file providing the researcher with a valuable bank of row data.

3. The learners' model in L.E.C.G.O.

The learner model of a problem solution in L.E.C.G.O. consists of a combination of the following five sub-learner models (SLM) using: 1) G.R.S., 2) F.T.R, 3) I.T.R., 4) P.C.R. and 5) P.L.C. Learners can use some or all RS above to produce their solution strategies to the given problems. Possible learner paths across RS are diagrammatically presented in Figure 1.

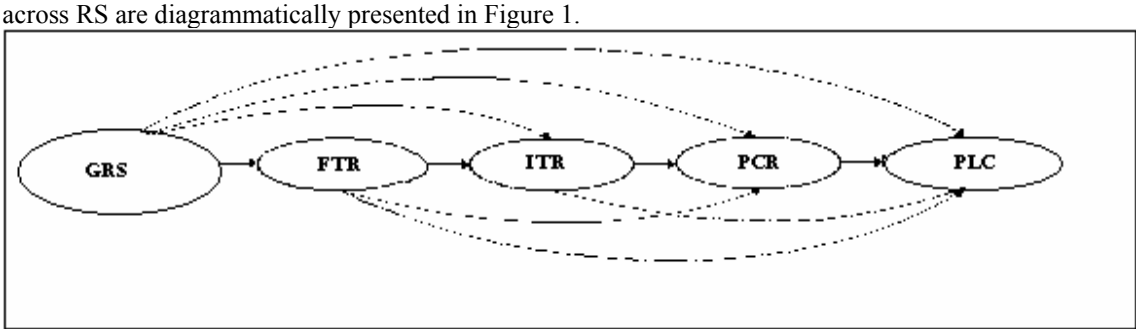

Figure 1. Possible learner paths in solving problems using L.E.C.G.O.M.R.S.

Learners can select from among these RS those most appropriate for their individual variety in order to produce a solution to a given problem. So, eight possible different paths (Pi, i=1,…,8) can be con-

structed. These paths are: P1) GRS->FTR->ITR->PCR->PLC, P2) GRS->FTR->PCR->PLC, P3) GRS->PLC, P4) GRS->FTR->PLC, P5)GRS->ITR->PCR->PLC, P6) GRS->PCR->PLC,and P7) GRS-> ITR->PLC.

In the following section, the specific learner models previously referred to are presented.

1.The learner model using G.R.S. Possible learner problem solution strategies (GRSi, i=1,…,3) to the given problems are: GRS1) Use of GRS to produce a complete graphic solution, GRS2) Use of GRS to produce an incomplete but acceptable graphic solution, and GRS3) No use of GRS.

2.The learner model using F.T.R. Possible learner problem solution (F.T.R.Si, i=1,…,4) strategies to the given problems could be constructed by using F.T.R. to produce: FTRS1) a complete solution, FTRS2) a vague description of the solution, FTRS3) a clear yet incomplete solution. In addition, students could construct strategies without the use of F.T.R (strategy FTRS4).

3.The learner model using I.T.R. Possible learner problem solution strategies (I.T.R.Si, i=1,…,9) to the given problems could be constructed by using the specific expressions provided to produce: ITRS1) a complete solution taking into account the appropriate coordinates, ITRS2) a complete solution taking into account the wrong coordinates, ITRS3) a partially correct, yet incomplete solution taking into account the appropriate coordinates, ITRS4) a partially correct, yet incomplete solution taking into account the wrong coordinates. In addition, students could form their own expressions in the imperative to produce four types of strategies similar to those mentioned above, namely: ITRS5, ITRS6, ITRS7, ITRS8. Students could also construct solution strategies without the use of I.T.R. (strategy ITRS9).

4.The learner model using pseudo-code (PCR). Possible learner problem solution (P.C.R.Si, i=1,…,9) strategies to the given problems can be constructed by using the provided algorithmic structures to produce: PCRS1) a complete solution taking into account the appropriate coordinates, PCRS2) a complete solution without using any coordinates, PCRS3) a partially correct, yet incomplete solution taking into account the appropriate coordinates, PCRS4) a partially correct, yet incomplete solution without using any coordinates. In addition, students could construct basic algorithmic structures by themselves and produce four types of strategies similar to those previous mentioned, namely: PCRS5, PCRS6, PCRS7, PCRS8. Students also could avoid using P.C.R in their strategies (strategy PCRS9).

5.The learner model using the programming language C (PLC). Possible learner problem solution (PLCSi, i=1,…,9) strategies to the given problems using the programming language C are: PLCS1) Constructing a complete program by selecting both the appropriate algorithmic structures in C and its appropriate graphical functions to produce a complete solution, also taking into account the appropriate coordinates. The programs run. PLCS2) As in PLCS1, with the exception that the produced solution is partially correct, yet incomplete. PLCS3) As in PLCS1, with the exception that the students use wrong coordinates. PLCS4) As in PLCS2, with the exception that the students use wrong coordinates. All programs could be constructed by using the last three strategies (strategies PLCS2, PLCS3 and PLCS4) run but do not provide the appropriate solution. PLCS5) Constructing a complete program by using both wrong algorithmic structures in C and wrong graphic functions at the same time as using correct coordinates. PLCS6) As in PLCS5, with the exception that the students use wrong coordinates. PLCS7) Constructing an incomplete program by using both wrong algorithmic structures in C and wrong graphic functions while at the same time using correct coordinates. PLCS8) As in PLCS7, with the exception that the students use wrong coordinates. All programs could be constructed by using the last four strategies do not run. PLCS9) Do not construct any program using P.L.C.

It is worth mentioning that, when learners act in the context of a specific RS, they have the opportunity to select, among the variety of possible specific solution strategies (presented in the previous section of this paper), the most appropriate strategy for their individual variety. All the previously mentioned solution strategies referred to a specific RS can be used in a variety of combinations following the main paths (Pi, I=1,..7) described in Figure 1. On the whole, learners have the possibility of expressing their individual variety in solving the given problems by following in total two hundred ninety nine (299) different specific solution paths: a) eighteen (18) different successful paths, b) eighty one (88) partially correct, yet incomplete paths, and c) two hundred non-successful paths.

4. The learner model in the paper & pencil environment and in Turbo C

It is worth mentioning that the learner model in both the paper and pencil environment and Turbo C is the same model presented above using PLC.. In these environments, the only successful strategy is PLCS1. There are three partially correct, yet incomplete strategies (PLCS2, PLCS3 & PLCS4) and five non-successful strategies (PLCS5, PLCS6, PLCS7, PLCS8 & PLCS9).

5. Conclusions

This paper presented the role of MRS in the enhancement of the learner model for the learning of programming in C. More specifically, an open learning computer environment (The L.E.C.G.O. environment) was designed, including a number of RS such as: a)Graphic, b) Text-based, c) Imperative text-based, d) Pseudo-code e) Programming in C and f) Graphic output of the written programs in C. Students can construct a variety of solution strategies for the given problems using these RS. In particular, learners have the possibility of expressing their inter-individual variety when solving problems by following in total two hundred ninety nine (299) different solution paths: a) eighteen (18) successful, b) eighty one (81) partially correct, yet incomplete, and c) two hundred (200) non-successful solution paths. It is worth noting that, both in the paper and pencil environment and in a typical compiler of Turbo C, learners have the possibility of solving the given problems by following in total nine (9) specific solution paths: a) one (1) successful, b) three (3) partially correct, yet incomplete, and c) five non-successful solution paths.

References

[1] K., Zikouli, M. Kordaki & E. Houstis, A multiple representational Environment for Learning Programming and C. The 3rd IEEE International Conference on Advanced Learning Technologies, Athens, Grece, p. 459, (2003).
[2] C. Janvier, Problems of representation in teaching and learning of mathematics. London: Erlbaum, (1987).
[3] M. Kordaki, The effect of tools of a computer microworld on students' strategies regarding the concept of conservation of area. Educational Studies in Mathematics, **52**, pp. 177-209, (2003).
[4] J. J., Kaput, The Representational Roles of Technology in Connecting Mathematics with Authentic Experience. In R. Biehler, R. W. Scholz, R. Strasser, B., Winkelman (Eds), Didactics of Mathematics as a Scientific Discipline: The state of the art (pp. 379- 397). Dordrecht: Kluwer Academic Publishers, (1994).
[5] D.H, Jonassen, Computers in the classroom: Mindtools for Critical Thinking. Colombus OH:Prentice Hall (1996).
[6] M., Kordaki & D. Potari, A learning environment for the conservation of area and its measurement: a computer microworld. Computers and Education, **31**, pp. 405-422, (1998).
[7] J-M., Laborde, & R.Strasser, Cabri-Geometre: A microworld of geometry for guided discovery learning. ZDM, **5**, pp. 171-177, (1990).
[8] M., Satratzemi, V., Dagdilelis & G. Evaggelidis, An Alternating Approach of Teaching Programming in The Secondary School. In Proceedings of 3rd Panhellenic Conference with International Participation, 'Information & Communication Technologies in Education', pp. 289-298, Phodes, Greece, (2002).
[9] R.Brooks Towards a Theory of the Cognitive processes in Computer Programming. Int. J. Human-Computer Studies, **51**, pp.197-211, (1999).
[10] L.E., Winslow, Programming Pedagogy. SIGCSE Bulletin, Vol. **28**, No. 3, pp.17-22, (1996).
[11] K. A., Lemone, & W., Ching, Easing into C: Experiences with RoBOTL. ACM, SIGCSE Bulletin, Vol. **28**, No. 4, 45-49, (1996).
[12] E., Soloway & J.C. Spohrer, Studying the novice programmer. Hillside, N.J. Erlbaum, (1989).
[13] C., DiGiano, K., Kahn, A. Cypher & D.C., Smith, Integrating Learning Supports into the Design of Visual Programming Systems. Journal of Visual Languages and Computing, **12**, pp..501-524, (2001).
[14] P., Brusilovski, E., Calabrese, J., Hvorecky, A., Kouchirenko & P., Miller, Mini-languages : a way to learn programming principles. Education and Information Technologies, **2**, pp.65-83, (1997).
[15] S. N., Freund & E.S., Roberts, THETIS: An ANSI C programming environment designed for introductory use. ACM, SIGCSE '96 **2**/96, Philadelphia, PA USA, pp.300-304, (1996).
[16] L., Vygotsky, Mind in Society. Cambridge: Harvard University Press, (1978).
[17] von Glasersfeld, E. An Exposition of Constructivism: Why Some Like It Radical. In R. B. Davis, C. A. Maher, and N. Noddings (Eds), Constuctivist views on the teaching and Learning of Mathematics (pp. 1-3). Reston VA: N.C.T.M., (1990).
[18] J., Piaget, B., Inhelder & A., Sheminska,The child's conception of geometry. NY:Norton & Company, (1981).