Using animated interactive analogies in teaching basic programming concepts and structures

Dimitrios Doukakis[1], *Grammatiki Tsaganou*[2], *Maria Grigoriadou*[3]

[1]*Interdisciplinary Program on Basic and Applied Cognitive Science, University of Athens, Greece, doukakis@di.uoa.gr*

[2,3]*Department of Informatics and Telecommunications, University of Athens, Greece, gram@di.uoa.gr, gregor@di.uoa.gr*

The use of analogies presented with interactive animation is considered as a possible way to deal with student's difficulties in understanding important programming concepts and structures. In this paper we present the development of three learning objects that use interactive animated analogies about the variable concept and the value assignment command, the conditional structures and the looping structures. A possible teaching approach based on exploratory activities is discussed. Furthermore data from a pilot experiment with the learning object about the variable concept are presented. This pilot experiment was designed to measure differences in students' performance resulting from their engagement in exploratory activities with the animated interactive analog. In this experiment we compare two instructional interventions with different versions of the learning object with or without an animated interactive analogy.

Keywords

Analogies, CS1 programming, Exploratory activities, Interactive animation, Learning Object.

1. Introduction

Various studies of high school novice programmers have described difficulties and misconceptions in understanding important concepts and structures such as the concept of variable as defined in introductory programming languages like Basic and Pascal [5], [6], [7], [12], [17], [18] conditional structures [5], [6], [11], [13], [15] and looping structures [5], [6], [11], [13], [14], [15].

The educational approach presented in this paper, for dealing with misconceptions and difficulties, exploits the use of educational analogies. The use of analogies by teachers, as an effort to help students understand complex science concepts, is quite common [4] and this also applies in teaching programming concepts and structures. An analogy is a mapping between similar features of dissimilar concepts, principles or formulas [3]. When the analogy is used for instruction the known concept is called analog (source) and the concept to be taught target. The term "model" sometimes is used instead of the term "analog" especially for dynamic systems. The term "analog" is preferred because as Glynn [8] states, " an analog is a kind of model but not all models are analogs"

According to Curtis's and Reigeluth's classification of analogies in written text [3], the relationship between the analog and the target can be either structural that focuses on the structure of a concept, or functional that focuses on the function or structural – functional. The presentation of the analogies can be verbal or pictorial or verbal-pictorial. Verbal, pictorial and verbal-pictorial presentation is adequate for structural analogies but is not so effective for presenting functional or structural-functional analogies.

Most programming concepts and structures are dynamic in nature since they are related with the execution of a program. As a result, analogies that are appropriate for teaching programming concepts and structures are functional or structural-functional analogies.

In this study a different way to present educational analogies is being studied. Analogies are presented in an animated form and they have interactivity features allowing students to control the execution of the animation. Furthermore the use of analogies was combined with exploratory activities.

2. Learning objects with interactive animated analogies

A Learning Object (LO) is an entity that can be used for learning. It is small, independent, reusable in different contexts, and can be part of a bigger learning entity [9], [19].

For the purpose of this study three digital learning objects for teaching important programming concepts and structures were developed. The first LO is about the concept of the variable and the value assignment command (variable LO), the second about the conditional structures (conditional LO) and the third about the looping structures (loop LO).

All three LOs consist of three parts:

1. Theory part. A part where theoretical issues of the each concept or structure are presented (variable concept and value assignment command, conditional structures, looping structures). The function and the syntax of the associated commands are presented in this part
2. Example part. A part with examples of using the relevant concept or structure .The examples are presented using the analogy
3. Activity part. A part where students can engage in activities. This part also makes use of the analogy.

Students are able to visit each part in the order of their choice by means of a navigation toolbar.

The educational analogies used in these LOs where simple mechanical analogies that have been chosen carefully to ensure that students are familiar with the analog and that the analog shares a big number of common attributes with the target.

In this study, “GLOSSA” [1] an educational Pascal-like programming language that follows the mini-languages approach [2] has been used. In “GLOSSA” the value assignment operator is the symbol “←” and all reserved words are in Greek (translated in English in this paper).

2.1 Exploratory activities

The activities designed for use with the animated analogies, follow the “Explorations” approach [10]. “Exploration” is a specific kind of homework assignment for novice programmers. Students must read a program, answer questions about the program, and make predictions about the program’s behaviour. Then students are asked to run the program, compare the predictions with the actual output of the program and give plausible explanations for wrong predictions.

In this study one value assignment command or an `IF` or a `WHILE` structure are given to the students instead of small programs.

In an exploratory activity with the variable LO, students can enter an assignment command, enter values of their choice for the variables included in the assignment command, predict the outcome of the command and then “execute” the command to see the real output.

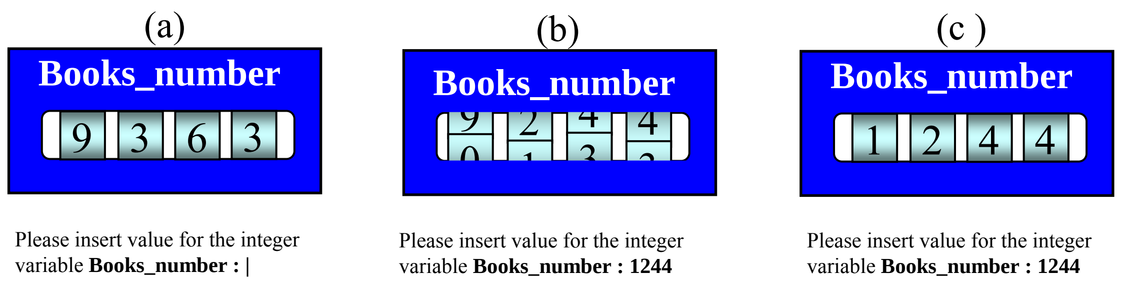
In the IF and the WHILE commands LOs students are able to give values to the variables in the logic expression. Values for these variables where proposed in the activity sheet and students are asked to predict the results from the execution of the commands. Then they are able to execute the command, observe the results and compare the results with their prediction. Students are also asked to write an explanation for any differences between their prediction and the execution results.

2.2 LO for the variable concept

2.2.1 Analog

Analogies commonly used for teaching the programming variable concept in procedural languages are the box or drawer analogy and the plate analogy. These analogies have been accused for being misapplied by students resulting in further misconceptions [5].

A different analogy for the programming variable was introduced in this study. It consists of a number of rotating cylinders with digits, characters or Boolean values depending on the type of the variable it represents (figure 1). It was expected to be familiar to the students from computer games (slot machine type) and television advertisements.



(a)
Books_number
9 3 6 3
Please insert value for the integer variable **Books_number :** |
(b)
Books_number
Please insert value for the integer variable **Books_number : 1244**
(c)
Books_number
1 2 4 4
Please insert value for the integer variable **Books_number : 1244**

Fig. 1: Snapshots of animated analogy used for an integer variable (a) initial state (b) while cylinders are rotating after entering value and pressing enter (c) final state

2.2.2 Interaction

A value can be given by students to the variable represented by the analog, with a value assignment command. In the value assignment command the student can enter a constant value or a formula containing variable names, constants and operators.

After entering the value to the value assignment command the cylinders of the analog are rotating and they stop to the digits that correspond to the new value.

Students are able to use a small number of variables of every type (integer, real, string, Boolean) already declared with predefined names. The number of the declared variables is restricted to three, in the prototype LO but larger number can be easily achieved if needed.

The declaration of the variables is visible to the students while engaging in the activity.

2.3 LO for simple conditional structure

2.3.1 Analog

The analog used for the simple conditional structure (`IF (logic expression) THEN … )` command is a system of pipes that form two possible routes for a falling ball (figure 2). The ball is directed to one of the two possible routes by means of a racket. Pipe routes correspond to the flow of control in the execution of the simple IF command while the movement of the racket is determined by the value of the expression used in the `IF` command. The straight alternative route corresponds to the case where the logic expression is false. The other route corresponds to the true case and the commands after `THEN` are executed. Commands before and after `IF` are executed anyway since they correspond to a part of the pipe system where only one route is available for the ball. The movement of the racket is controlled by the evaluation of the logic expression used in the `IF` command. The racket directs the ball to the appropriate route.

2.3.2 Interaction

Students can enter a logic expression and commands of their choice before, inside and after the `IF` command. Students are able to use a small number of variables (integer or real) with predefined names. The number of available variables is restricted to three, in the prototype LO but larger number can be easily achieved if needed. The current value of these variables is visible to students while engaging in the exploratory activities.

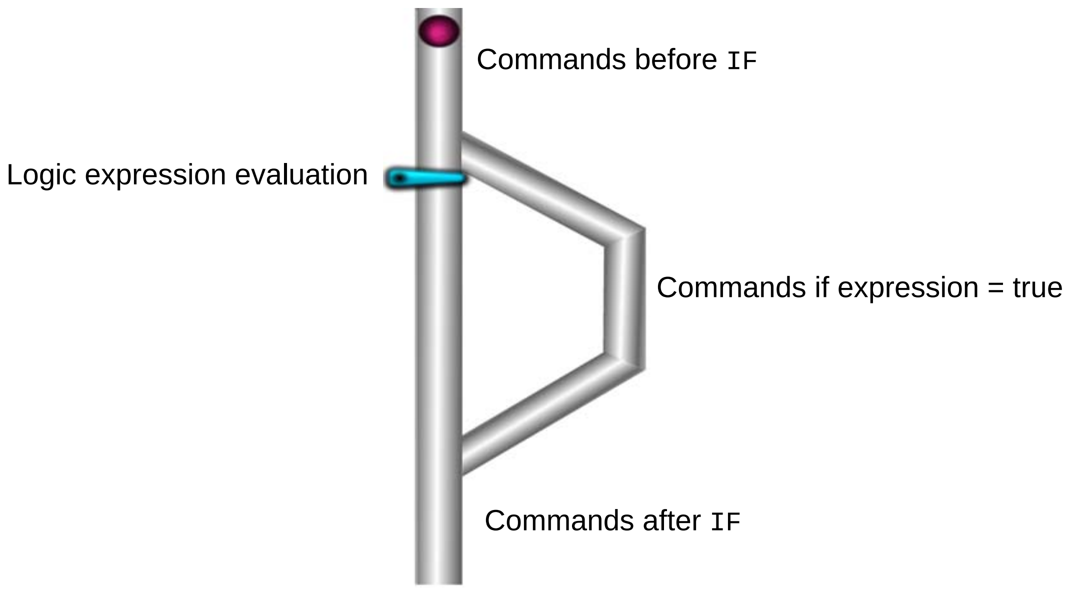


Figure 2. Analogy used for **`IF…THEN`** command

2.4 LO for looping structure

2.4.1 Analog

The analog used for the `WHILE` command (`WHILE (logic expression) DO`) is a system of pipes that form two possible routes for a falling ball (figure 3). The ball is directed to one of the two possible routes by means of a racket. Pipe routes correspond to the flow of control in the execution of the `WHILE` command while the movement of the racket is determined by the value of the logic expression used in the command. The straight alternative route corresponds to the case where the logic expression is false. The cyclical route corresponds to the true case and the commands inside the `WHILE` are executed. The ball continues to follow the cyclical root for as long as the logic expression is true. Commands before and after `WHILE` are executed anyway since they correspond to a part of the pipe system where only one route is available for the ball. The movement of the racket is controlled by the evaluation of the logic expression used in the `WHILE` command. The racket directs the ball to the appropriate route.

2.4.2 Interaction

Students can enter a logic expression and commands of their choice before, inside and after the `WHILE` command. Students are able to use a small number of variables (integer or real) with predefined names. The number of available variables is restricted to three, in the prototype LO but larger number can be easily achieved if needed. The current value of these variables is visible to students while engaging in the exploratory activities.

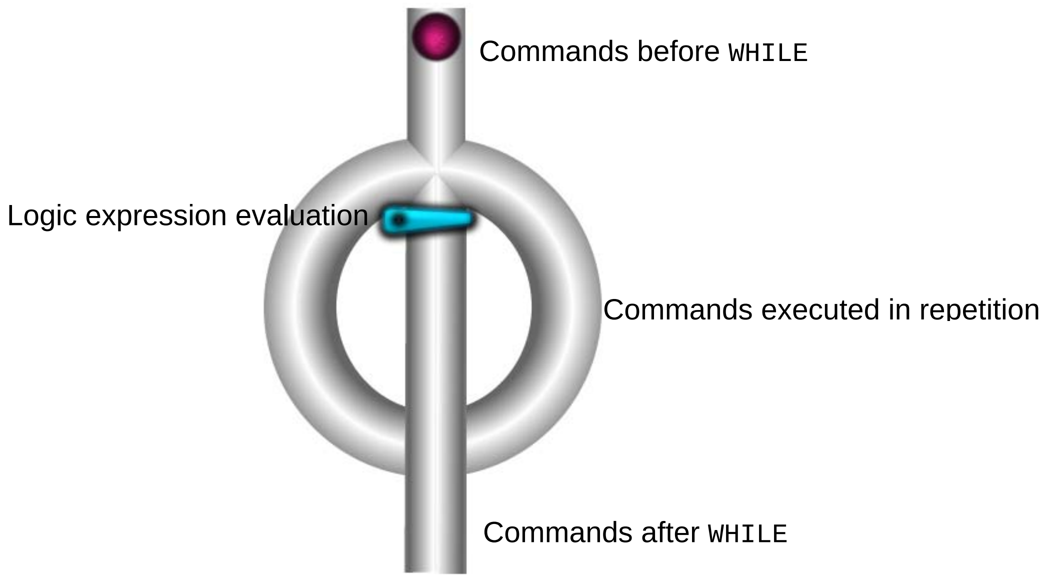


Figure 3 Analogy used for **`WHILE…`** command

3. Method

In order to measure the effectiveness of the analogy use in this educational approach, pilot experiments were designed with all three LOs. Students' understanding of each concept is measured using questionnaires for misconceptions detection on each of the three concepts (variable-value assignment command, conditional structures, repetition structures.)

The hypothesis tested in these experiments is whether students using the interactive animated analogy for exploratory activities have an increased performance in questionnaires designed to measure their understanding and diagnose their misconceptions.

Other research questions concern students' familiarity with the selected analog as well as their interest in experimenting with it.

So far only the pilot experiment with the variable LO has been carried out. Following subsections (3.1, 3.2, 3.3) and section 4 refer to this experiment.

3.1 Materials

- A version of the LO that included the interactive animated analogy and a version of the LO that had the same parts and the same interactivity features in the activity and did not use the animated analogy.

- An activity sheet with exploratory activities as described in 2.1.
- A questionnaire for misconceptions detection. The questionnaire (see appendix) included nine open type questions used in previous studies [5], [7], [18], [19]. A justification of the answer was also requested. Two more questions where also added to indicate student's familiarity with the analogy (Q10), and interest level on a Likert scale (1-5) for using the analogue (Q11).

3.2 Participants

Thirty-three introductory programming students from the last three grades of a high school (aged 15-17 years) in the Athens area participated in this study. All of them have been taught the programming variable concept by their teacher in a traditional instruction without analogy use.

Two introductory programming classes with similar grades in the programming course formed the two groups of the experiment. Twenty-one students took part in the analogy group and twelve in the control group.

3.3 Procedure

Both groups where provided with activity sheets and engaged in the exploratory activities without any time restriction.

Then both groups answered questions Q1-Q9. Questions Q10 and Q11 where answered only by the analogy group. Questions Q1-Q9 were marked with 0 for wrong answers and 1 for right answers. Answers to questions that included explanations where considered correct only when both the answer and the explanation where correct.

3.3 Results

The differences in performance were found to be in favour of the analogy group for all the 9 questions of the questionnaire but it was statistically significant at 0.05 level for 4 (Q4, Q5, Q6, Q7) of the questions (see table 1). Furthermore the analogy group students found the analog "familiar" (83%), "interesting above average" (38%) and "very interesting" (62%).

Table 1 : Independent samples t-tests for analogy and control groups

Question	**T- test**
Q1	T(31)=1,544, n.s.
Q2	T(31)=1,544, n.s.
Q3	T(31)=1,397, n.s.
Q4	T(31)=2,328, p=0,027
Q5	T(31)=2,436, p=0,021
Q6	T(31)=3,798, p=0,001
Q7	T(31)=3,521, p=0,001
Q8	T(31)=1,812, n.s.
Q9	T(31)=3,015, p=0,005

4. Discussion

This pilot experiment revealed that engaging novice programmers in exploratory activities with the use of a properly selected interactive animated analogy can help them deal with some of their misconceptions in the concept of programming variable.

A possible explanation is that the experimentation with the interactive animated analogy, help students identify important features of the programming variable concept that in a conventional instruction remained less obvious.

These features relate to the:

- Function of the value assignment command (Q4)
- The difference between the name and the value of a variable (Q5)
- Storage capacity of a variable in terms of size and precision (Q6, Q7)

A qualitative analysis of students' answers is also planned in order to give some evidence-based explanation for the differences in performance between questions.

The experiment also indicated that students were familiar with the analog chosen and interested on experimenting with it.

Subsequent experiments with all three LOs are planned to measure the individual effects of the combined factors (analogy, animation and interaction).

Analogies are well known for helping students understand complex science concepts but they are also well known for causing misconceptions in cases where students over generalize and map not corresponding features of source and target concepts [16]. So an important question has to do with the problems that instruction with analogy is known to cause when carried too far. In subsequent experiments questions will be added to detect new misconceptions that can derive from misapplication of the specific analogy.

Another problem concerns students' familiarity with the analog proposed by the teacher, the textbook writer or the educational software designer. Unfortunately there is not always available an analog familiar to all students, that is also appropriate for the instruction of a specific concept. Therefore the above approach cannot be proposed as an instructional suggestion to all cases of difficult programming concepts.

However, it is important to note that computer based simulations and animations give a new and perhaps more efficient way to incorporate analogies in instruction.

5. Appendix : Questionnaire for misconceptions detection on the concept of the variable in "GLOSSA"

Q1. Where are programming variables been stored?

Q2. What will be printed on the screen? Give an explanation of your answer.

```
VARIABLES
INTEGER: D
PRINT D
…………..
```

Q3. Is the following assignment command correct? Give an explanation of your answer.

```
VARIABLES
CHARACTER: A,B,C
……………..
A ← B+C
```

Q4. What will be printed on the screen? Give an explanation of your answer.

```
VARIABLES
INTEGER: A ,B,C
…………..
B← 5
C← 3
A← B+C
PRINT A
```

Q5. What will be printed on the screen? Give an explanation of your answer.

```
VARIABLES
CHARACTER: A
……………..
A← “B”
PRINT A
```

Q6. Is the following assignment command correct? Give an explanation of your answer.

```
VARIABLES
INTEGER: A
……………..
A← 175688909897687563254357657899066890098867675564
```

Q7. Is the following assignment command correct? Give an explanation of your answer.

```
VARIABLES
REAL: A
………………..
A←3,1456…. (infinite number of decimal digits)
```

Q8. Is the following assignment command correct? Give an explanation of your answer.

```
VARIABLES
INTEGER: A
……………..
A← “ATHENS”
```

Q9. What will be printed on the screen? Give an explanation of your answer.

```
VARIABLES
INTEGER: A
…………..
A← 5
A← 7
PRINT A
```

Q10. Have you seen before the analog used (the rotating cylinders) ?

Q11. Did you find it interesting working with the analog?
1. Not interesting at all 2. Below average interest 3. Average interest 4. Above average interest 5. Very interesting

References

1. Bakali, A. Giannopoulos, I. Ioannides, C. Koilias, C. Malamas, K. Manolopoulos, I. Politis P. (1999), “Application Development in Programming Environment”, Athens , Ministry Of Education and Religious Affairs – Pedagogic Institute.

2. Brusilovsky, P., Calabrese, E., Hvorecky, J., Kouchnirenko, A., and Miller, P. (1997) Mini-languages: A Way to Learn Programming Principles. *Education and Information Technologies* **2** (1), 65-83.

3. Curtis, R. Reigeluth, C. (1984) The use of analogies in written text Instructional Science,13 , 99-117.

4. Dagher Z. R. (1995) Review of Studies on the Effectiveness of Instructional Analogies in Science Education, Issues and Trends, *Science Education*, 79(3), 295-312.

5. Du Boulay, B. (1989), Some difficulties of learning to program, In E. Soloway & J. C. Spohrer (Eds), *Studying the Novice Programmer*, Hillsdale, NJ, Lawrence Erlbaum Associates , 283-299.

6. Ebrahimi, A. (1994), Novice programmer errors: language constructs and plan composition, Int. J. Human-Computer Studies, 41, 457-480.

7. Fesakis, G. Dimitrakopoulou, A. (2005) Cognitive Difficulties of Secondary Education Students relating to the Concept of Programming Variable, 3rd Panhellinic Conference “Didactics of Informatics”, Korinthos, Greece.

8. Glynn , S.M. Britton, B. K. Semrud-Clikeman, M. & Muth, K.D. (1989). Analogical reasoning and problem solving in science textbooks. Handbook of creativity. New York: Plenum, pp. 383-398

9. IEEE Learning Technology Standards Committee (LTSC) (2001) *Draft Standard for Learning Object Metadata Version 6.1.* http://ltsc.ieee.org/doc/

10. Lischner, R. (2001), Explorations : Structured Labs for First-Time Programmers, Proceedings of the ACM SIGCSE ’01 Conference, Charlotte, USA, 154-158.

11. Putnum, R. T. Sleeman, D. Baxter, J., Kupsa, L. (1989), A summary of the misconceptions of high school BASIC programmers, In E. Soloway & J. C. Spohrer (Eds), *Studying the Novice Programmer*, 301-314, Hillsdale, NJ, Lawrence Erlbaum Associates.

12. Samurçay, R. (1989), The concept of variable in programming: Its meaning and use in problem solving by novice programmers, In E. Soloway & J. C. Spohrer (Eds), *Studying the Novice Programmer, 161-178,* Hillsdale, NJ, Lawrence Erlbaum Associates.

13. Sleeman, D. Putnum, R. T. Baxter, J. Kuspa, L. (1988), An introductory Pascal class : A case study of student’s errors, in R.E. Mayer (Ed.). Teaching and learning Computer Programming : Multiple Research Perspectives, Hillsdale, NJ: Lawrence Erlbaum Associates, 237-257.

14. Soloway, E., Bonar, J. & Ehrlich, K. (1983), Cognitive Strategies and Looping Constructs: An Empirical Study, Comm. of the ACM, 26(11), 853-860.

15. Spohrer, J. and Soloway, E. (1986) Analyzing the high frequency bugs in novice programs**,** Workshop on Empirical studies of programmers, 230 – 251.

16. Thagard, P. (1992) Analogy, explanation and education. Journal of Research in Science Teaching, 29, 537-544.

17. Tzimogiannis, A. Komis, B. (2000) The concept of variable in Programming: student’s difficulties and misconceptions , in Komis, B, (eds) Procedings of the 2nd Panhellinic Conference “The Technologies of Information and Communication in Education”, 103-114, Patra, Greece.

18. Tzimogiannis, A. Politis, P. Komis , B. (2005) Study of the Representations that Last Year High School Students have for the Concept of Variable, 3rd Panhellinic Conference “Didactics of Informatics”, Korinthos, Greece.

19. Wiley, David. A. (2002) "Connecting Learning Objects to Instructional Design Theory: A Definition, a Metaphor, and a Taxonomy." *The Instructional Use of Learning Objects* (Bloomington, IN: Agency for Instructional Technology)