Virtual Student's Bar: Educational Software for the Novices' Learning of Iteration Algorithmic Structures

Maria Kordaki
Dept of Cultural Technology and Communication, University of the Aegean
e-mail: kordaki@cti.gr

Vasilios Kalavrouziotis
Dept of Computer Engineering and Informatics, University of Patras
e-mail: kalabrouzi@ceid.upatras.gr

Abstract: Multiple representational educational software can assist students in their learning process in many ways. The main purpose of this paper is to present the design and implementation of an educational, multi-representational environment [named Virtual Student's Bar (VSB)] that is dedicated to help novices understand the basic concept of algorithms and iteration statements, by using everyday life's motivational activities like the process of creating a drink. The design of VSB is based on constructivist theories of learning. Students can virtually create step-by-step a drink among a number of drinks that are available in the application. In that way, students are more likely to understand new concepts such as procedural algorithmic logic and iteration structures, at the same time combining fun with learning. VSB also provides the students opportunities to experiment with multiple representations such as natural language, pseudo-code and flow-chart techniques using icons familiar to them.

Keywords: Educational software, multiple representation systems, basic algorithmic iteration structures, constructivist learning theories, secondary education

Introduction

Nowadays, computer environments are a part of our lives. Computers have also great influence in the learning process since more and more educational software products have been designed and developed last years. So, students have to be familiar with this new type of software in order to overcome some learning difficulties and to be helped to understand some science and computer concepts in an easier way. Psychological studies of computer programming expertise have approved that turning a beginner into an expert needs a four year program (Winslow, 1996). In a survey of program understanding, it was mentioned that expert programmers: have organized and specialized knowledge schemas; use both general problem solving strategies and specialized strategies; use a top-down approach to decompose and realize algorithms (Robins, Rountree & Rountree, 2003).

Three decades ago, the prevalent opinion about how a student learns, was that the knowledge was passively gained from the teachers. However, these traditional opinions have been criticized by modern theories of learning suggesting that in those traditional approaches knowledge is transferred to students with little concern weather the student understands or assimilates the subject of learning (Leidner & Fuller, 1999). In fact, knowledge and communication are not transferable commodities (Jonassen, 1999). Thus, in contradiction to the traditional learning methods in the west world, constructivist theories of learning were developed. According to constructivism, a teacher must not ignore the student's already existing knowledge (Ben-Ari, 1998). The positive role of authentic real life activities on students' learning was also acknowledged (Jonassen, 1999).

Appropriately designed educational software can support constructivist learning in various ways (Jonassen, 1999). However, such educational software design should take into consideration the ways with which students learn (Squires & Preece, 1999). To this end, real life activity based educational software could supply students with the motivation to deal with such activities, with ultimate goal to understand some concepts and meanings in a specific scientific field (Kordaki, Miatides, Kapsabelis, 2008). Hands-on experience educational software that provides students with opportunities to acquire hands-on experience with basic aspects of the concepts in question is appropriate for the learning of all learning subjects and especially for the learning of computer algorithms (Kordaki, Miatides, Kapsabelis, 2008). However, hands-on experience is not enough. Engaging learners with Multiple Representation Systems (MRS) is also essential (Johnson, Moher, Ohlsson & Leigh, 2001). In fact, the use of MRS is a way to make learning process more attractive and more efficient to students (Ainsworth, 2006). Some students seemed to understand better a concept when it is represented in some types of representations like hypermedia. However, natural language or diagrammatic representations are more understandable for other learners. In addition, if an educational software presents a plethora of representations, students can work with the representation of their personal choice and cognitive development. Research on learning with multiple representational systems has proved that when students can interact with a suitable representation their performance could be enhanced (Kordaki, 2003). So, a piece of educational software must represent its concepts with more than one representation (see Ainsworth, 2006). Even though more and more educational computer environments are designed, it is noticed a lack of multi-representational applications which target to teach basic algorithmic structures like iteration statements. Among the benefits of using MRS, also, are that they can support new and different ideas and that they support learners with deeper understanding of a learning subject (Ainsworth, Bibby & Wood, 1997). Even, if there are different representations to a task, none of this careful design is likely to matter unless students process in an active way the information which is given by the software (Goldman, 2003).

As regards the learning of basic programming constructs by novices, they seem to find serious difficulties, as it has been reported by a number of researchers (Soloway and Spohrer, 1989; Winslow, 1996; Robbins, Rountree and Rountree, 2003; Hadjerrouit, 2008; Eckerdal, 2009). In fact, novices seem to encounter serious problems to understand the problems at hand, needless to say that they seemed to be puzzled by the semantic and the syntactic rules of the programming language in use. In addition, beginners seemed to use general problem solving techniques rather than techniques that are more relevant to the algorithmic solutions necessary for problem solving relevant to a computer language (see Winslow, 1996). Most importantly, novices did not fully realize how a computer works.

In our attempt to provide novices with opportunities to learn basic programming constructs as the iteration structures, we designed and implemented a specific piece of educational software entitled 'Virtual Student's Bar' (VSB). For the design of this software, modern theories of learning and the significance of the use of MRS on students learning within the context of enjoyable and everyday activities as the preparation of drinks that could take place in a Virtual Student Bar were taken into account. This is the contribution of this paper. In the next section of this paper, the design, the specific features and the technical characteristics of the proposed computer learning environment are introduced. Next, a summary and some proposals for future work are presented.

The General Architecture of Virtual Student's Bar

The target of this work is to design and implement a web-based application, for educational purposes, in order to help primary and junior high school students to understand the concept, the operation and the usage of iteration structures in computer algorithms. The result of our work is a software product and we refer to it here as Virtual Student's Bar. For the design of VSB it was considered appropriate to provide students with opportunities to combine learning with fun in a user friendly environment (MacFarlane, Sim & Horton, 2005). It is critical for children to interact with an educational software in a playful way, without realizing that they learn (Barab, Thomas, Dodge, Carteaux and Tuzun, 2005). It was also considered that students had no previous knowledge about the general idea of what algorithms are and of basic algorithmic statements, like assignment statements. For that reason, various representation systems were used such as: natural language, flowcharts -in the form of pictures, which are familiar to the students- and pseudo-code. The essential role of MRS has been suggested for the learning of basic algorithm concepts by novices (Kordaki, Miatides, Kapsabelis, 2008). Finally, it was considered, that nowadays, many children in Secondary Education are familiar with the use of a personal computer.

Opening the Virtual Students Bar (VSB) application, a student is ready to interact with the educational software. The general architecture of VSB application consists of the following four parts: Part A: Home page, Part B: Theoretical issues, Part C: Readymade examples, Part D: 'Do it yourself' part that gives students the freedom to experiment with the idea of algorithms with and without iteration statements. Each student is able to visit and use each of these parts with the help of a navigation menu that is located on the left side on each screen. This menu is accessible from any VSB's page, in order to assist users for an easier navigation in the application. In the following paragraphs, the aforementioned parts are briefly presented.

Part A: Home Page. This is the first page of the application. Here, one can see a picture, relative with the application and the following salutation statement *'Welcome to Virtual Student's Bar an educational web-based software that aims help novices realize basic concept about algorithms and iteration structures'*.

The general purposes of the VSB application are also mentioned in brief in this page. For usability issues, each part of the general architecture is displayed in the navigation bar. Thus, the "theory" link is placed on the top of the navigation menu. As a second link in this menu the "Examples" link is placed. By clicking on this link readymade examples of algorithms using iteration structures are presented. Then, the "Do It Yourself" link is placed, where students can try –even using the trial and error method- all the knowledge they acquired through their study of the theoretical issues and the examples presented in the aforementioned links, until a positive event will be occurred. Such positive event within VSB is to successfully create a drink by following the right sequence of a number of given steps. Students can also directly visit the "Do It Yourself" link and produce such positive events by experimenting with the features provided at the same time using the 'Theory' and the 'Examples' links as scaffolding elements to their experimentation.

Part B: Theoretical issues. In this part, theoretical issues about Algorithms and iteration structures are presented. This theoretical part is divided into the following five parts:

Part B.1: *Basics about what an algorithm is*. In this part, the basic concept of algorithm is simply presented through an every day example, so that it could be understandable by young students. To this end, examples of algorithms of our everyday life are used such as the production of a glass of orange juice. The focus here is to help students to clarify what the problem is that have to be solved by an algorithm; that is to make a glass of orange juice in this case. The next step is to encourage students to make it clear that this can be done by a sequence of actions each of which should be performed in one step (the algorithmic solution). It is also mentioned that this sequence of steps is not unique. To support students understand what an algorithm is, appropriate examples are provided.

Part B.2: *Multiple representation systems for representing algorithms.* Here, four different ways for algorithm representations were stated and analyzed, namely: natural language, diagramming techniques such as flowcharts, and coding using pseudo-code or any of computer programming languages. To help students understand the aforementioned representation systems appropriate examples are presented.

Part B.3: *Diverse types of iteration structures*. In this part, the necessary knowledge about the three basic iteration structures is presented, namely: do-while, while-loop, and for-loop iterations structures. Having already provided with the general concept of what an algorithm is, the presentation of a specific algorithmic statement, the iteration statement to the students was considered appropriate to be presented here. To provide students with opportunities to better understand the aforementioned iteration structures appropriate examples are demonstrated.

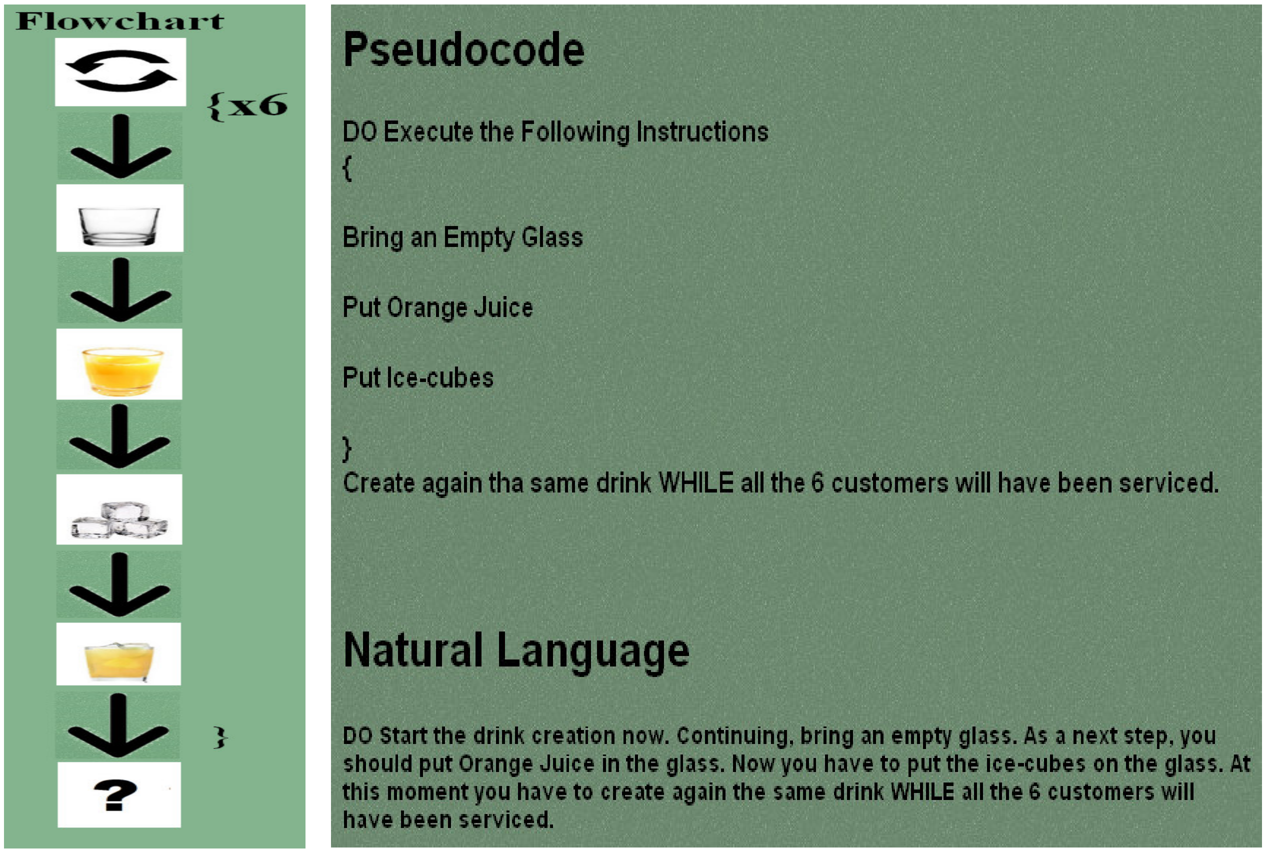
Part B.4: *Differences between diverse iteration structures.* Here as well, the basic differences between the three different types of iteration statements are briefly mentioned. The purpose is to provide learners with appropriate knowledge about when each iteration structure is suitable and when it is not. Students are supplied by appropriate examples to be scaffold in their understanding of the differences among the aforementioned iteration structures.
It is worth noting that, each of the aforementioned theoretical parts can act as a help for the students and it is briefly presented to avoid boredom by the young learners (see Squires & Preece, 1999).

Part C: Readymade Examples. When students click the 'Readymade Examples' link, they are transferred on a page where they are asked to choose between some readymade examples of problem solving using algorithmic structures. Some of these examples represent simple algorithms where iteration statements are not used. These algorithms are referred to the description of the necessary steps in order to produce a drink, as for example, to fill a glass with water and ice cubes. In this example, it is underlined that from our everyday life the only obvious first step of such an algorithm is to bring an empty glass. Then, one can put in it either the water or the ice cubes. In that way, an attempt has been made to help novices to understand that some steps in an algorithm must go before some others, in order to reach a desired result. In other examples, the three types of iteration structures are used. All examples are represented in MRS such as natural language, flow - chart with appropriate pictures and pseudo-code.

Part D: Do It Yourself. Constructivists suggest that learning is more effective when a student is actively engaged in the learning process rather than attempting to passively receive knowledge. In this part each student is able to create –by using hands-on experience- the recipes of various drinks in the form of different algorithms. In fact, students can choose from a drop down menu what kind of the drink they like, so that to proceed in the formation of its recipe as an algorithm within the context of VSB. Students can also choose –from another drop down menu- how many times the process of a drink's creation will be repeated. Next, they have to choose –from a third drop down menu the representation system they prefer to use in order to form the appropriate algorithm using specific iteration statements. Specifically, students can form the appropriate recipes by using the representation system they prefer, namely: natural language using specific reserved expressions; flow-chart using appropriate figures familiar to the student and pseudo-code. The description of an algorithmic solution using the previously mentioned flow-chart is decided to support beginners to bridge their intuitive –visual- knowledge with the 'new' knowledge that is represented in the form of pseudo-code. Natural language is also more understandable to someone who is beginner, so it is used to throw bridges across student previous knowledge and the new knowledge that students gain from the use of VSB. Natural language also assists learners to translate the information from other types of representations. On the whole, by using the aforementioned three representations of an algorithmic process, students can connect their visual intuitive knowledge with the symbolic knowledge represented in natural language as well as with a knowledge represented in pseudo-code that is a form compatible with the way a computer operates.

All the three types of iteration statements can be used, so a student can choose to create, for example, five glasses of orange juice with the do-while iteration statement or seven glasses of fruit punch with the use of the for-loop structure. In each student's action, a feedback step is always available. If a student does not follow the correct step towards a successful result, then a message advises her/him that their action was wrong without revealing the correct answer. During their experimentation, learners have the choice to view, at any time, the theoretical part, which is given as help, in order to be assisted if they wish to. When the learner manages to successfully create a drink using a specific representation system, then, she/he has the chance to see the algorithm for drink creation in the rest two representation systems. That was decided to help students overcome their difficulties to translate between representations (Ainsworth, 2006). As students can construct different algorithms to create a drink, they can experiment different solutions to the task at hand. In this way they can enhance their knowledge about the implied concepts that is acknowledged as essential in constructivist learning (Kordaki, 2003). It is worth noting that, VSB has the appropriate knowledge to evaluate all possible correct solutions in the three representation systems mentioned above. Finally, it is expected that learners should be motivated to be engaged with the aforementioned experimentations in order to form algorithmic solutions for drink creation within the context of VSB. To this end, it is suggested that students should deal with environment which represents motivation tasks (Jonassen, 1999).

An Example of Possible Use of VSB: Here, let's act as a student who tries to create a drink of her/his preference. In fact, she/he has the choice to select the drink who prefer (a glass of water with ice cubes, iced coffee, orange juice, fruit punch, and pineapple-orange juice), and then, the number of glasses of the selected drink (the available number of choices is between 1-10 number of glasses). Let's assume that she/he wants to create a few glasses of orange juice, e.g. six glasses. Initially, she/he has to select from a dropdown menu the orange juice option. In a next step, he has to choose the specific number of glasses, which are six in our example. Finally, the learner should choose the appropriate for her/him representational system: the pseudo-code for the purposes of our example. When the learner has completed the aforementioned steps, she/he will be moved to a new screen, where the specific instructions of the recipe for the selected drink are presented to the student in a random order. So, she/he has to put them in the correct order. This can be happened by clicking an instruction each time. If the instruction is put in the correct order then a picture relevant to this instruction is appeared. On the other hand, if the student puts an instruction in a wrong order, then it is appeared a message that informs him that the step, which she/he chose, is not correct. Only when a student successfully creates her/his drinks, then a link is appeared, which drives the student to another page, where she/he can view the process of their drink creation in the rest two representations: natural language and flow-chart in our example (Fig.1).



Pseudocode

```
DO Execute the Following Instructions
{

Bring an Empty Glass

Put Orange Juice

Put Ice-cubes

}
Create again tha same drink WHILE all the 6 customers will have been serviced.
```

Natural Language

DO Start the drink creation now. Continuing, bring an empty glass. As a next step, you should put Orange Juice in the glass. Now you have to put the ice-cubes on the glass. At this moment you have to create again the same drink WHILE all the 6 customers will have been serviced.

Figure 1. A snapshot illustrating the 3 representations of a possible drink creation by a student

VSB has been implemented using Php 5.3.5 and HTML4 and XHTML and it runs under a WAMP server 2.1.

Summary and Plans for Future Work

A web-based educational software application, which is called Virtual Student's bar (VSB) was presented in this paper. This application targets on helping novices to understand the general concept of algorithms and of basic algorithmic structures, namely: iteration structures. VSB's environment has been designed by taking into consideration modern theories of learning and novices' difficulties in learning programming. With the assistance of VSB, students can be introduced to new concepts such as procedural algorithmic logic, as well as to the use of repletion structures in programming by exploiting their existing knowledge from their everyday life activities. In fact, students can realize and understand the aforementioned concepts by experimenting with real life activities, like drink creation. Furthermore, VSB provides students with a number of multiple representational systems such as natural language using specific reserved expressions; flow-chart using appropriate figures familiar to the students and pseudo-code. By working within VSB environment, learning process could become more efficient, attractive and motivating to learners than the school-like, paper and pencil, boring and meaningless context. For the future, we plan to integrate within VSB additional real life activities except for drink creation as well as more algorithmic structures. We also plan VSB to be tested and be evaluated by a number of real students in a classroom.

References

Ainsworth, S. (2006). DeFT: A conceptual framework for considering learning with multiple representations. *Learning and Instruction*, 16, 183-198.

Ainsworth, S., Bibby, P. & Wood, D. (1997): Information technology and multiple representations: new opportunities – new problems, *Journal of Information Technology for Teacher Education*, 6:1, 93-105.

Barab, S., Thomas, M., Dodge, T., Carteaux, R. & Tuzun, H. (2005). Making Learning Fun: Quest Atlantis, A Game Without Guns. *Educational Technology Research & Development*, Vol. 53, No. 1, 2005, pp. 86–107.

Ben-Ari, M. (1998). Constructivism in Computer Science Education. *SIGSCE 98* Atlanta, GA USA, 257-261.

Eckerdal, A. (2009). *Novice Programming Students' Learning of Concepts and Practise*. Dissertation presented at Mathematics and Computer Science, Department of Information Technology, Upsalla University, Sweeden, March, 6, 2009 available at http://uu.diva-portal.org/smash/record.jsf?pid=diva2:173221.

Goldman, S. (2003). Learning in complex domains: when and why do multiple representations help? *Learning and Instruction*, 13 (2003) pp. 239–244.

Hadjerrouit, S. (2008). Towards a Blended Learning Model for Teaching and Learning Computer Programming: A Case Study. *Informatics in Education*, 2008, Vol. 7, No. 2, 181–210.

Johnson, A., Moher, T., Ohlsson S. & Leigh, J. (2001). Exploring Multiple Representations In Elementary School Science Education. *Proceedings of the Virtual Reality 2001* Conference (VR.01).

Jonassen, D. H. (1999). Designing constructivist learning environments. *Instructional design theories and models*, 2, 215-239.

Kordaki, M. Miatidis, M. and Kapsampelis, G. (2008). A computer environment for the learning of sorting algorithms: design and pilot evaluation. *Computers and Education*, Vol. 51(2), pp. 708-723.

Kordaki, M. (2003). The effect of tools of a computer microworld on students' strategies regarding the concept of conservation of area. *Educational Studies in Mathematics*, 2003; 52: 177-209.

Leidner, D.E. & Fuller, M.A. (1996). Improving Student Processing and Assimilation of Conceptual Information: GSS-Supported Collaborative Learning vs. Individual Constructive Learning. 29th *Hawaii International Conference on System Sciences (HICSS). Volume 3: Collaboration Systems and Technology, 1996.* Maui, HI, USA, pp.293.

MacFarlane, S., Sim, G. & Horton, M. (2005). Assessing Usability and Fun in Educational Software. *IDC2005*, pp.103-109, Boulder, CO, USA.

Robins, A., Rountree, J. & Rountree, N. (2003). Learning and Teaching Programming: A Review and Discussion, *Computer Science Education*, 13:2, 137-172.

Soloway, E. & Spohrer, J. C. (1989). *Studying the novice programmer*. Hillside, N.J. Erlbaum.

Squires D. & Preece, J (1999). Predicting quality in educational software: Evaluating for learning, usability and the synergy between them. *Interacting with Computers*, 11 (1999) 467–483. Elsevier Science B.V.

Winslow, L.E. (1996). Programming Pedagogy -- A Psychological Overview. *SIGCSE BULLETIN.* Vol. 28 No. 3 Sept. 1996.

Acknowledgements

We would like to acknowledge Leonidas Papadopoulos who assisted us in the implementation of VSB.