TEACHING OBJECT-ORIENTED THINKING TO NOVICE PROGRAMMERS USING THE AGENTSHEETS ENVIRONMENT

Spyropoulos Charalampos
MSc in Computational Intelligence
B.Sc. Informatics Engineering
Ph.D. Candidate Department of Educational and Social Policy
University of Macedonia,
Thessaloniki, Greecee

Vassilios Dagdilelis
Assistant Professor
Department of Educational and Social Policy
University of Macedonia
156 EgnatiaStr., GR-540 06,
Thessaloniki, Greece

Georgios Evangelidis
Assistant Professor
Department of Applied Informatics
University of Macedonia
156 EgnatiaStr., GR-540 06,
Thessaloniki, Greece

ABSTRACT

Java and Visual Basic are the most commonly used programming languages in teaching programming to beginners. The advantage is that students use currently dominant programming tools in the market. However novice programmers have problems of comprehension, at least this is reported by the relevant research, and this is due to the complexity of the professional programming development environments for Java and Visual Basic. Moreover, modern programming models, like the object-oriented scheme, are posing a series of additional intellectual obstacles to beginners. Many researchers have suggested special environments for dealing with those problems, specifically designed for introductory teaching. In the present article, the AgentSheets programming environment is proposed for an introduction in object-oriented programming. The AgentSheets environment can have a positive effect in developing the object-oriented thinking for novice programmers. Visual and tactual embedded features, as well as interactivity and feedback capabilities in all development and execution phases, in addition to rapid example implementation are able to successfully interact in the improvement of algorithmic rationale and object-oriented thinking. Therefore, AgentSheets introduces a wide range of educational uses.

KEYWORDS

Objects-First, micro-languages, object-oriented programming, novice programmers, AgentSheets

1. INTRODUCTION

During the last decade, the ongoing shifting to object-oriented languages in the software development arena has led many academic establishments in the adoption of the "Objects-First" approach for the introductory teaching of programming (ACM 2001). Besides, in many countries the object-oriented model has been adopted by secondary education (e.g. Greece) up to a certain extent. Still, it is realized that the adoption of this model and the use of object-oriented languages delivers to novice users substantial problems in understanding and learning, for example, the lack of using concepts before their implementation, the concurrence of class and objects, message passing, confusion of state and behaviour etc. (Holland et al, 1997). Professional programming languages (Java, Visual Basic, C++, C#), although they possess the advantage of being the ones that are to be eventually used by students when working in the market (Allen, 1996), can actually become quite problematic to novice programmers. Apart from classical problems attributed to the nature of a professional tool (Integrated Development Environment - IDE) like the extended language, the professional (and not educational) orientation, and the nature of the applications that one can implement as a beginner (Brusilovsky et al. 1997), there are a series of educational obstacles having to do with the objected-oriented philosophy, when the latter constitutes the model of an introductory approach to programming. "Micro-languages" (Brusilovsky et al, 1997) like Logo (Papert, 1980), Karel (Pattis, 1981), Marta (Calabrese, 1989), language subsets like Blue J (Kölling, M. and Rosenberg, J., 1996), and also educational environments like *MicroWorlds Pro®* have been proposed by many researchers for dealing with these problems. In Section 2, we discuss the problems that are usually met when teaching object-oriented programming, and in Section 3, we present the concepts behind micro-languages and Object-Orinted Programming. Section 4, deals with the AgentSheets environment and its object-oriented features as well as the "Object-First" teaching approach it follows. Finally, Section 5 concludes and describes our future plans.

2. PROBLEMS IN TEACHING OBJECT-ORIENTED PROGRAMMING

Many of the aforementioned problems regarding the understanding and learning procedures originate from some more general obstacles when the object-oriented philosophy constitutes a model of introductory approach to programming. Below, we list some of them:

- The starting lessons, which are the ones that introduce students to the basic principles of programming, use examples of reduced complexity that do not outline the benefits of object-oriented design and program building.
- Object-oriented languages require knowledge of basic programming structures and a variety of language characteristics and capabilities prior to the accelerated object development (Cooper, Dann and Pausch, 2003).
- Visual representation and monitoring of objects is a rather sophisticated procedure that demands prior programming and management knowledge of the environment (Proulx, Raab and Rasala, 2002).

So, the effort for comprehension and correct expression utilizing such professional programming languages in the case of novice programmers, leads students to over-focus in language syntax, rather than concentrate in planning and developing object-oriented design skills and algorithmic thinking.

3. "MICRO-LANGUAGES" AND OBJECT-ORIENTED PROGRAMMING

Most of the micro-languages that have been constructed and have been applied until today focus on the procedural or functional paradigm of programming. Some of them, in their latest versions, support characteristics and structures of the object-oriented paradigm of programming, like StarLogo (Resnick M., 1994), Karel++ (Bergin, J., Stehlik, M., Roberts, J., and Pattis, R., 1997) and *MicroWorlds Pro®* without necessarily having a pure object orientation. A new generation of unmixed objected-oriented environments have been developed over the last few years and are now in the final stage of their research application and usage as development tools centralized around programming thinking and introduction to programming. Some relevant examples of those environments are Objectkarel (Xinogalos, 2003), ALICE (Pausch et al, 1995), that using a very rich in structures and commands object-oriented language and a three dimensional virtual world addresses mainly neophyte programmers of higher education, and AgentSheets (Repenning, 1995). Creation of simulations in the AgentSheets environment is primarily aiming to the evolvement of object-oriented thinking and lesser in familiarizing with the commercial programming languages. Although the languages that are used in such environments usually comprise of a subset of a commercial programming language and do not always support all the programming structures, they ought to provide characteristics very similar to the ones that exist in the fundamentals of the object-oriented model which is originally targeted. Fundamental features of object orientation, as enumerated by Alan Kay (Alan C. Kay, 1993), are:

1. Everything is an object.
2. Every type of calculation is carried out by the objects, which communicate with each other by stating requests through message passing.
3. Every object is an instance of a class of objects. This class defines the characteristics of a set of uniform objects.
4. Inside classes behaviour of objects, which belong to this class, is described. Therefore, each object that belongs to a class can execute this behaviour.
5. Class are organised in a tree-wise hierarchical structure that determines the inheritance of characteristics from parent classes.

Those features outline the distinctive marks an object-oriented programming language, which are of course extended and analysed into a majority of additional characteristics that are also differentiated depending on the language used. A micro-language has to encompass and outline in the clearest possible way the above characteristics in order to be able to function as a tool of object-oriented development and allow a smooth, and free from misunderstandings, transition to a commercial language.

4. AGENTSHEETS ENVIRONMENT AND ITS OBJECT-ORIENTED FEATURES

The AgentSheets environment is a tool for creating simulations of multiple applications that allows non-programmers to create their own simulations and games. It is mainly a visual programming environment that combines agents, spreadsheets and Java scripting tools into an integrated platform. The Agents are presented as small moving images in the computer screen and they have associative behaviours that specify their own interaction with the environment. Those behaviours are defined by the end-user, thus, allowing users with any programming background to create agents with composite behaviours (Ioanidou A, Spyropoulos C., Viglas L., 2005). Programming in AgentSheets is done through the use of the Visual AgentTalk (VAT) programming language. The essential features of this language and the simulation development pattern with regard to the adoption of the basic object-oriented characteristics are described in the following section.

4.1 Essential Features of VAT

According to Alan C. Kay's formulation, the essential features of the VAT language that follow the object-oriented rules are, respectively:
Agent Gallery: The collection of classes (agents) that are created by the programmer are the unique entities of the language. All the rest of the objects, which can exist and be programmed in the contracted simulations, are created based on these classes.
Object Behaviour: Objects carry all of the calculations and it is not possible for code to exist without belonging to an object's class behaviour. Furthermore, objects have the ability, via a set of commands like BROADCAST, MAKE, HEAR, ON *MESSAGE*, to set forth requests using message passing.
Object Memory: Apart from the typical variables, of local and global type that objects may refer to, objects are able to also relate to objects or classes via the visual process and during the programmatic stage.
Object Creation: Objects are created either dynamically – during code execution with the use of the CREATE command issued by another object – or by the user, before or during the program execution.
Hierarchical structure: The Visual AgentTalk language allows the one level hierarchical structure used for the inheritance of parental characteristics and the sole differentiation allowed is the graphical representation of a new class.

4.2 Simulation Development

In addition to its language characteristics, AgentSheets adapts to the object-oriented rationale with another equally important feature: its interface environment and the tools it supplies for the development of applications.
Object-Oriented Development: In the AgentSheets program development environment everything is an object. The classes of objects are not predefined; they are user-defined and are responsible for their representation and their naming behaviour. The behaviour of a class is built through the VAT programming language. Every object class owns its own behaviours that are organised into methods, are described by a Trigger, and consist of rules like "If…then…" on which the programmer assigns the suitable conditions and actions.
Object-Oriented Language Structure: The fundamental features are the TRIGGERS, the CONDITIONS and the ACTIONS that spring from already existing libraries. Yet, the above language features are objects that can be manipulated via the user interface. The language object flow (conditions and actions) at the user interface is the characteristic that attributes the concept of "Tactile Programming" to VAT language. The

meaning of "tactility" is used here by the same way as Papert (Papert, 1993) used it in order to explain the programmer's familiarity degree in accordance with her computational objects (Alexander Repenning & Ambach 1996; Alexander Repenning & Ioannidou, 1997). The principles and the programs of Tactile Programming not only promote visual representations, which help in readability and comprehension of programs, but also are dynamic and include manipulations, like the parameters' definition through the use of visual fields. (Figure 1)

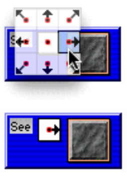

Figure 1. Parameters' definition through the use of visual fields

Object-Oriented Debugging: In the AgentSheets environment there is no room for syntactical errors, since the procedure of code typing does not take place, allowing this way the programmer to worry merely for the logical errors. The debugging tools provided by the environment are control of an action, a condition, a rule even a complete method, directly on an object with immediate visual and auditory feedback during code execution.

4.3 From Design to Teaching

In the "Object-First" teaching approach, there exists an objective teaching trouble in the definition and description of the object-oriented rationale as a model for thinking and problem solving. On this effort, many books attempt to introduce the student to the fundamental concepts of object-oriented thinking by using examples from the real world, very similar to everyday life. Through the use of this kind of examples, that exploit the already established knowledge and experience of the students, the building of new representation structures and the realization of semantic changes is much more easily performed (Komis, 2005). Many times though, these examples are impossible to be programmatically implemented, at least during the initial programming steps, since their implementation would require an extended knowledge and experience in programming languages. Such an example, which has been used by introductory books in programming, is the example of watching television. Deriving from the student's everyday life is really able to by all means signify the basic principals of object-oriented world. Television, even though its internal components are irrelevant to us, still, is an object that supports a set of predefined behaviours. The basic points that should be stressed and used in a relevant example would be:

- The channel selection is something that the television internally may support, only if it accepts the suitable request from another object, in our case the remote control. The request is executed taking under consideration the possible parameter that might be the channel number or simply the previous-next channel.
- The same applies for the sound as well; the requests to which the TV responds are fluctuation of sound, activation or deactivation of sound, change of auditory filter. All of these are preset behaviours of a specific TV appliance.

By extending and analysing examples of this kind we are able to describe several of the basic characteristics of the object-oriented model. An implementation of the example though, and the development of a simulation program in the computer supplies both the ideal and integrated feeling and the perception to the students of what it really means to create object-oriented programs. With the VAT language of AgentSheets students are able to materialize and see in action such an example in a short period of time. What's more, the change and the formulation of different scenarios involves them in a process of conversion and reestablishment through which they comprehend not only the concepts of object orientation, but also the basic programming rationales, like software life-cycle, local and global variables, constants, conditional execution, etc.
Realization in AgentSheets follows the following stages:
Definition of objects and their representation: In the preceding example, objects are the television set and the remote control. Their various representations are the picture of different channels that shows the screen for the TV object, and for the remote control's object, the several buttons on it.

Next, every object's behaviour is defined by using the high level structures provided by VAT, such as ON (message), BROADCAST (message), CHANGE (depiction), PLAY SOUND (sound) and all these inside sentences IF…THEN… (Figure 2)

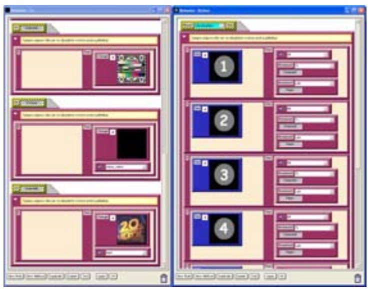

Figure 2. Object's behaviour is defined

In every development stage, the student is able to examine the effect of every rule on objects he puts on the work sheet and to have immediate feedback by controlling and re-establishing the way that any behaviour is developed. Finally, via the execution of the simulation (Figure 3) he controls the overall result and the way various objects interact with each other as well as with the simulation user.

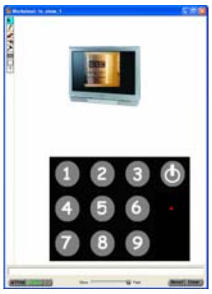

Figure 3. Execution of the simulation

With the completion of the simulation several work field alteration scenarios can be proposed, as for example the addition of one more similar class object, i.e., another one television that simultaneously interacts to the requests of the same remote control, automated search and channel programming, etc., thus, outlining throughout experimental procedures that the students themselves implement fundamental programming concepts of object-oriented programming paradigm.
The benefits of using AgentSheets during the implementation of such an example are:

- Visualisation of example
- The students are able to bring ideas to life from the very first lessons and experience the joy of creation.
- The capability of experimenting and learning through experience.

5. FUTURE RESEARCH - EXPERIMENTS

It is estimated that the programming environment of AgentSheets may successfully affect the object-oriented thinking development of novice users. Apart from the benefits that are provided by the procedure of modelling and simulations in various cognitive areas, it may help in the introductory programming lessons and operate as a lever for growing programming abilities, especially as far as novice users are concerned. Through the visualisation in all levels, the immediate feedback during the software development process, and with the ability to resolve and simulate problems of high complexity, it can encourage and stimulate new programmers in parallel to the development process of their basic programming capabilities. The research question is whether AgentSheets can function as a teaching environment for programming and constitute the "AlgorithmLand" in accordance to the "MathLand" that Papert suggests in his book "Mindstorms". The visual and tactility characteristics that unify the environment in the programming language level as well as the interaction and feedback capabilities in all development and execution phases of the programs, may positively assist the development of algorithmic thinking and object-oriented rationale. Our future plans, include the development of introductory and advanced targeted courses using AgentSheets and their experimental application in secondary education schools and higher education IT departments, aiming in this way record and analyse the advantages that might derive both in the conceptual and the basic programming capabilities level of students. While there are many programming environments of this kind that could be used as introductory programming tools, a single and perfect choice is not possible. Nevertheless, their combination with other micro-worlds and micro-languages or even more the addition of new features to them can lead to better results. We believe that the use of this kind of environments, especially at the higher education level can function as a bridge for a smooth and successful transition to commercial object-oriented programming environments.

REFERENCES

ACM (2001), *Computing Curricula 2001*, Computer Science, Final Report, (December 15, 2001),The Joint Task Force on Computing Curricula, http://www.computer.org/education/cc2001/cc2001.pdf

Alan C. Kay (1993), "The early History os Smalltalk", *The Second ACM SIGPLAN History of Programming Languages Conference (HOLP-II)*, ACM SIGPLAN Notices 28(3): 69-75.

Allen, R.K., Grant, Douglas D., & Smith, R. (1996). "Using Ada as the first Programming language: A Retrospective". In *Proceedings of Software Engineering: Education & Practice, (SE:E&P'96)*, IEEE Computer Society Press.

Bergin, J., Stehlik, M., Roberts, J., and Pattis, R. (1997). *Karel++ A Gentle Introduction to the Art of Object Oriented Programming*, Wiley.

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

Calabrese, E. (1989) Marta - the "Intelligent Turtle". *Proceedings of Second European Logo Conference, EUROLOGO'89* . Edited by G. Schuyten and M. Valcke. Gent, Belgium, 30 Aug – 1 Sep 1989, pp. 111-127.

Cooper S., Dann W., Pausch R., (2003), "Teaching Objects-first In Introductory Computer Science", *SIGCSE'03* February 19-23, 2003, Reno, Nevada, USA.

Holland Simon, Robert Griffiths, Mark Woodman (1997), "Avoiding Object Misconceptions", *SIGCSE '97 CA*, USA

Ioanidou A, Spyropoulos C., Viglas L. (2005), «Η Ελληνική έκδοση του AgentSheets ως περιβάλλον διδασκαλίας προγραμματισμού μέσα από διαθεματικά παραδείγματα»,*ETPE*, Σύρος 13, 14, 15 Μαΐου 2005

Komis I. Βασίλης (2005), *«Εισαγωγή στη Διδακτική της Πληροφορικής»*, Εκδόσεις Κλειδάριθμος, σελ. 222

Kölling, M. and Rosenberg, J. (1996) An object-oriented program development environment for the first programming course. *Proceedings of 27th SIGCSE Technical Symposium on Computer Science Education*, ACM, Philadelphia, Pennsylvania, 83-87, March 1996.

MicroWorlds Pro® [computer software]. (2000). Logo Computer Systems Inc., Highgate Springs, VT. [Online]. Available: http://www.microworlds.com/solutions/mwpro.html

Papert, S. (1980*). Mindstorms: Children, Computers and Powerful Ideas*, Harvester Press, Brighton, Sussex.

Papert, S. (1993). *The Children's Machine*, Basic Books, New York

Pattis, R. E. (1981). *Karel - the robot, a gentle introduction to the art of programming* . Wiley,London.

Pausch, R. (head), Burnette, T., Capeheart, A.C., Conway, M., Cosgrove, D., DeLine, R.,Durbin, J., Gossweiler, R., Koga, S., & White, J. (1995). Alice: Rapid prototyping system for virtual reality. *IEEE Computer Graphics and Applications.*

Proulx, V., Raab, R., Rasala, R., (2002), Objects from the beginning – with GUIs, *Proceedings of th 7th annual conference on Innovation and Technology in Computer Science Education*,Archus, Denmark, June 2002, 65-69

Repenning, A. and Sumner, T. (1995). "Agentsheets: A Medium for Creating Domain-Oriented Visual Languages." *Computer 28*: 17-25.

Repenning, A., & Ioannidou, A. (1997). Behavior Processors: Layers between End-Users and Java Virtual Machines. Proceedings of the *1997 IEEE Symposium of Visual Languages*, Capri, Italy, pp. 402-409.

Repenning, A., and J. Ambach (1996), "Tactile Programming: A Unified Manipulation Paradigm Supporting Program Comprehension, Composition and Sharing," *Proceedings of the 1996 IEEE Symposium of Visual Languages*, Boulder, CO, Computer Society, pp. 102-109.

Resnick M. (1994). *Turtles, Termites, and Traffic Jams*, MIT Press

Xinogalos S. (2003), objectKarel: A Didactic Microworld for Teaching Object- Oriented, *ITiCSE'0, 03*, June 30-July 2, 2003, Thessaloniki, Greece ACM 1-58113-672-2/03/0006.