Dual Instructional Support Materials for introductory object-oriented programming: classes vs. objects

Susana Montero, Paloma Díaz, David Díez and Ignacio Aedo
Laboratorio DEI. Computer Science Department
Universidad Carlos III de Madrid
{smontero,pdp,ddiez,aedo}@inf.uc3m.es

Abstract— **Visualizations are widely used in programming courses but the integration of these software tools into classrooms is not straightforward since there is a lack of information about the real benefits for learning and how to combine them with traditional lectures. We describe in this paper an experiment of using dual instructional support materials (textual and visual programs) in the lab classes in a Java-based CS1 course. The approach consists of two phases: students work the class concept in a traditional programming environment and the object concept in a visual environment as Greenfoot. The experiment shows positive results in terms of concepts understanding in students with no prior programming experience. Therefore we also suggest this as a possible way for integrating visualizations to the course.**

Keywords-component; Object-oriented programming, Visualization, Novice Programmers, Greenfoot

I. INTRODUCTION

Many studies have been carried out to know firsthand the main problems and difficulties in object-oriented introductory courses (CS1) both from the teaching and the learning point of view [1][2].

In a CS1 using an object-oriented language, it is crucial that students get a good understanding of key abstract concepts like object and class at an early stage [3]. Since in our degree on Computer Science we follow an approach that combines a strong start in basic constructs with early object-orientation [4], we are concerned with this issue.

Novice programmers do not have a concrete model in their everyday life to handle with these abstract concepts [5]. According to [6] between 75% and 83% of students are visual learners and, therefore, we could provide interactive visualizations for the students to concretize abstract concepts at the beginning of the course. Some of these visualization tools are devoted to help students "see" what an object actually is [7], as BlueJ [8], Greenfoot [9] or Alice [10].

Encouraged by this trend, we decided to use one of these tools in a first semester course in object-oriented programming with Java in order to engage students and improve their comprehension of the class and object concepts. However, the introduction of this software tools into the teaching practice is not straightforward since there is a lack of information about the real benefits for learning [7].

Consequently, using a visualization tool also implies to design a teaching strategy to make the use of such tools as efficient as possible from a pedagogical point of view. We have adopted an incremental approach separating and grouping concepts and stages. The class concept and the compilation stage are worked in a Java development environment. The object concept and its behavior are worked in a visualization tool for Java like Greenfoot, in which students' programs can be run step-by-step or in continuous play mode without making changes in their source code. For that, we have developed several dual instructional support materials in order to work with the same examples in both environments.

In this paper, we describe the basis and motivation of our approach and present the experimentation developed to evaluate the impact of these dual material within our CS1 course.

II. MOTIVATION AND THE RELATED WORK

Research into novices' understanding of object oriented has revealed several misconceptions and problems that concern the notions of class and object [3] and sequence of method calls and their effects on objects [5]. According to that and our experience, students have problems in the first stages of learning to understand the difference between classes and objects and between methods and their invocations.

Visual environments for teaching elementary object oriented programming have been developed in order to provide a concrete model to students and they might handle these abstract concepts [11], but there is little research into the effectiveness of such tools [12]. Moreover, it has been reported that the utilization of mere educational tools is not the best strategy. Students should also get acquainted with the professional programming environments they will use when working as professionals [13].

In [14] researchers attributed students' misconceptions about the execution of a sequence of instructions to the lack of distinction between the environment and the execution of a program, because both the class definition and the object creation were made in the same environment without a main program class.

Since we are aware of the very inconsistent results showed in different research studies, our perspective is that the program visualization should be used as a complementary resource but

978-1-4244-6571-2/10/$26.00 ©2010 IEEE April 14-16, 2010, Madrid, SPAIN
IEEE EDUCON Education Engineering 2010 – The Future of Global Learning Engineering Education

CS1 students need also to face traditional programming environments.

In [15] researchers integrated the use of visualizations to students' preparation for exercise sessions as homework. Our study attempts to integrate visualizations during the lab classes in order to improve the aforementioned misconceptions.

III. THE TEACHING STRATEGY

In our classes, the learning resources are structured into didactic units about the fundamental concepts of programming and objected-oriented principles along with short exercises for further practice about the concepts. In lab sessions students develop these concepts by creating and writing programs using Java in the JGRASP[16] object workbench environment. One of the problems in traditional programming environments is that object behavior is not directly observable, but these environments allow students to get acquainted about general concepts relating to computation, such as source code, compilation, execution or declarations of fields and methods in the class. Complementarily, visual environments allow active interaction and experimentation with object instances and to explore behavior by dynamic access.

We argue that if students develop these concepts separately in two complementary environments (programming and visualizing), they will improve the understanding of them. Consequently we decided to develop dual instructional material that can be used both in a traditional programming environment and a visual environment. In the former environment, students will work the class concept and the compilation stage. In the second environment they will play their solutions in order to work object creation and invocation of a method on an object, and thus they will understand better the sequence, first object creation, after method call.

The overall objective is to provide students with additional support for the lab classes, allowing difficult abstract concepts to be illustrated and reinforcing more effectively, like the class and the object concepts.

Next, we will explain briefly the instructional support materials under study.

A. Instructional Support Materials

At the early stage of the course the class and the object concepts are worked through a typical early exercise where we model geometric figures. This domain is simple for students and it does not present an added difficulty in the domain knowledge.

In a first step, students are asked to represent the *Point* concept like a simple class whose attributes, *x* and *y*, are of Java's primitive types and getters and setters methods are defined. In a second step, two kinds of figures are introduced to show the collaboration between objects. The *Triangle* is a geometrical figure formed by three *points* and the *Circle* is formed by a centre point and the radius. Moreover, a UML representation of the classes is provided.

Students have to think about how to use the learned concepts to implement the solution. The development is done using jGRASP and the solution is seen in a visual environment.

IV. CREATING DUAL INSTRUCTIONAL SUPPORT MATERIALS

In order to create the visual instructional material, first we analyzed some of the most outstanding tools [11] and selected Alice [10], BlueJ [8] and Greenfoot [9]. All of them have its own website, books and papers that evidence that are well known and widely used. Moreover, these learning tools are recommended by Sun for users without programming experience [17].

During this analysis, we considered the following restrictions aimed at integrating the visual environment in the most efficient way in our CS1 course.

(i) Our students need to develop the basic java programming skills using proper syntax in order to accomplish assignments in this and other related courses. (ii) We cannot afford to spend a significant amount of class time teaching other environment, as well jGRASP, or event-driven concepts to support interactivity with GUIs. Indeed the main disadvantages of using visualization tools include the physical and cognitive overload required to use the tools both from students and teachers. It takes time to install and use the environment. (iii) The environment should allow us to represent visually the instances of classes described in our lab session material without changing the source code. Students have to experience how their code works to learn from their own mistakes. In the next paragraphs we describe the three tools we studied and how they fulfill or not our three requirements.

Alice (http://www.alice.org) is a 3D animation environment, developed at Carnegie Mellon University, in which objects and their behaviors are visualized. Its interface is based on drag-and drop manipulation of program components. Three-dimensionality provides a sense of reality for objects, but Alice hides the underlying language behind a "graphical" language, so it does not support teach text based programming.

BlueJ (www.bluej.org) is being developed and maintained at the University of Kent at Canterbury, UK, and La Trobe University, Melbourne. It provides a full Java environment in which the OO software project structure is presented graphically using UML-like diagrams. Users are able to create directly objects of any class by using icons, and then to interact with their methods. However, it does not display any visual clues about the object's state or behavior.

Greenfoot (http://www.greenfoot.org) was built by some of the same people who created BlueJ, so it is based on the Bluej platform. Greenfoot combines a Java IDE with a framework for producing Java programs that can be visualized in two-dimensional grid. This framework is based on a world environment with visual interactions between the world and the objects in it. Applications in Greenfoot are called scenarios.

TABLE I. VISUAL ENVIRONMENTS ANALYSED

	Alice	BlueJ	Greenfoot
Proper java syntax (i)	×Not text based programming	✓Full Java	✓Full Java
Java environment (ii)	×drag-and drop manipulation	✓Java IDE	✓Java IDE
Visual representation (iii)	✓3D	×icons	✓2D

Table I shows as the visual environments does cover (✓) or not (×) our restrictions to develop the visual instructional material. Greenfoot provides full java syntax, an integrated development environment, and 2D visual representation for our previous material.

Two Greenfoot scenarios were developed from the solution proposed for the exercises presented in the previous section (The point scenario and the geometric figures scenario). For each class, a visual class was created in the Greenfoot scenario. The aim was to minimize the student's contact with the way of programming in this environment, but providing concrete experiences with objects.

Fig. 1 shows a screen shot of the Geometric Figures scenario. The large grid area is called the "world" (1). In order to get a more real experience, a Cartesian coordinate system has been painted in which the geometric figures as objects will be created. The right side is the "class display" showing all Java classes involved in the scenario. From the bottom, the classes created by the students (2) are presented (Circle.java, Point.java and Triangle.java), and above them, the classes provided by the scenario (3) devoted to be a visual representation over the world (Circle_Greenfoot.java, Point_Greenfoot.java, and Triangle_Greenfoot.java). The way to interact with objects is through their methods showed in a context menu (4).

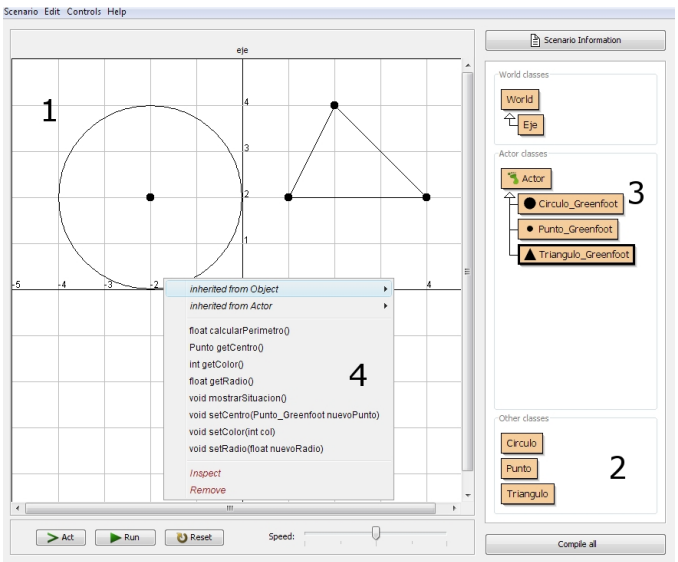

Figure 1. A screen shot of the Geometric Figures scenario

V. EVALUATION STUDY

We performed an evaluation study aimed at: (i) investigating the effect of using these dual instructional support materials in learning and (ii) examining whether these materials might contribute to better understand some basic object oriented concepts.

Two groups of students participated in the study. One group worked with the dual instructional material (GG, 15 students) and the other group just with the material for the jGRASP environment (CG, 18 students). All the students had roughly the same level of academic preparedness since those students with previous knowledge in Java were excluded from the study. Therefore, the concepts of object orientation in Java were new to all the participants.

Before the experimental sessions, the theoretical concepts were explained in common lectures. Two sessions were carried out to perform the scenarios described in the previous section. In both sessions the control group (CG) only used jGrasp whilst students in the other group (GG) were additionally asked to play their programs using Greenfoot. Sessions in this second group lasted a bit more since a briefing on the use of Greenfoot was required. Moreover, in their second session they were also asked to fill a questionnaire on the Greenfoot utility and usability as described below. Table II summarizes the experiment schedule.

TABLE II. EXPERIMENTAL PROCEDURE

Pre-experiment	Both	Lectures on theoretical concepts Class, methods, objects, instance, method call
Session 1	Both	Q1
	Both	Programming SC1 in jGRASP
	GG	Visualizing SC1 in Greenfoot
Session 2	Both	Programming SC2 in jGrasp
	GG	Visualizing SC2 in Greenfoot Q2
Session 3	Both	Q1

A. *Instruments for Data Collection*

We collected students' data and opinion using two types of anonymous questionnaires.

Table III shows the first type of questionnaire (Q1). It was aimed at identifying the level of understanding of concepts. It was filled for the first time at the beginning of the first session before performing the Point scenario (SC1). It consisted of a number of open questions where students described in her own words the concept of class, object, method, how an object is created and how methods are invoked. The students were not notified about the test in advance. This questionnaire was filled again at the end of the second session to measure students' progress in understanding concepts.

TABLE III. QUESTIONS FOR Q1

Evaluation Questionnaire - Q1
What is a class?
What is an object?
How is an object created?
What is a method?
How is a method called? Give an example if necessary.

The second questionnaire (Q2) was devoted to evaluate Greenfoot usability and its utility to improve the understanding of object oriented concepts, as demonstrated *in* Table IV. It was filled by students in the GG group at the end of the second session. This questionnaire had a number of closed questions in which a five-point Likert scale ranging from *none* (1) to *a lot* (5) was used to choose the response that best represented her opinion relative to features of the tool and scenarios. It also included a section with open questions where students were asked to express their opinions to enhance the dual instructional material and the experience.

TABLE IV. QUESTIONS FOR Q2

Evaluation Questionnaire – Q2
How easy is to use Greenfoot?
How easy is to visualize your exercise solution in Greenfoot?
How useful are the scenarios to understand the difference between class and object?
How useful are the scenarios to understand the creation of an object?
How useful are the scenarios to understand the state of an object?
How useful are the scenarios to understand the method call?
How useful are the scenarios to understand the collaboration between objects?
Would you like to visualize the rest of lab exercises in this environment?
How useful is the Point scenario to understand the object-oriented basic concepts?
How useful is the Geometric Figures scenario to understand the object-oriented basic concepts?
Do you think that the visualization is a good mechanism to understand better the object-oriented concepts or the jGRASP environment is enough?

B. Data Analysis and Results

For the sake of clarity we will structure the data analysis and results into two parts: the effects on learning outcomes and the students' perception about the usability and utility of the material.

Because we were interested in exploring whether there is a significant difference among groups before and after the experiment, we made an analysis based on the scoring of the students' answers for Q1 in a range from 0 to 10 points. Our hypothesis was that "if the groups could be considered equivalent in knowledge before the experiment, and after it they had significant differences, these differences could be attributed to the dual instructional material".

The data in Table V summarize the students' score for the questionnaire Q1, before and after the lab sessions. For each group we have identified the minimum and the maximum value for scores, the median and quartiles, and the standard deviation (SD) in order to have a first perception of the group equivalence.

TABLE V. MEASURES FROM Q1 SCORE

	GG (15 students)		CG (18 students)	
	Before	***After***	***Before***	***After***
Minimun	0	3	0	0
Maximun	6	10	7	9
Q1	1	6,5	1	4,25
Median	4	8	3	6,5
Q3	5	9	4	8
Mean	3,13	7,46	2,67	5,67
SD	2,13	2,16	2,25	2,68

Students from GG and CG found difficult to answer Q1 before the practical sessions, with a mean of 3.13 and 2.67 respectively. Since our groups are not formed by random assignment, an independent sample T-test was used to analyze the equivalence between GG and CG using the values of column before. This showed not statistically significant difference between GG and GC ($p= 0.5535$), so we can assume both groups were equivalent in knowledge before the experiment.

The column after shows Q1 score after finishing the second lab session. Students from GG increased their mean to 7.46 whereas CG only to 5.67. This shows a significant statistical difference between both groups ($p =0.0457$). Moreover, these results support the research made by [15] in which visualization was used to prepare the exercise sessions.

The Figure 2 shows in percentages the students' progress by ranges: a decrease in the grading (-10 to 0), an increase of three points (0 to 3) and an increase of more than four points (4 to 10). Most of students improved their grading after the practical work, but in GG even a 60% increased their grading in more than four points compared to a 44% percent in CG. Moreover, in GG no student decreased its grading whilst in CG a 6% got lower grades.

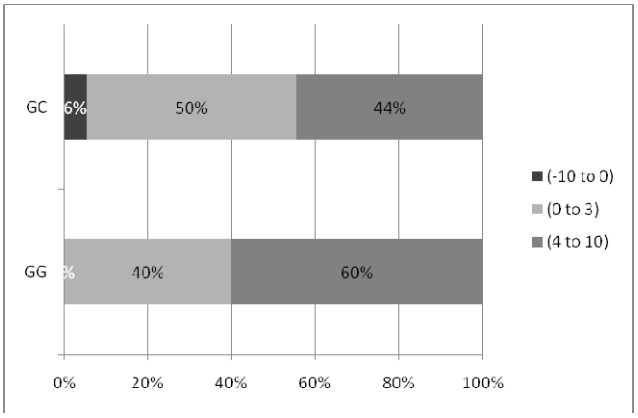

Figure 2. Percentages of the score improvement for Q1

Table VI presents the percentages of improvement for each concept asked in Q1. GG improves in almost all concepts, less in the concept of class. These data allow us to confirm the initial idea about the benefits of each environment. CG used just jGrasp in which the structure of class and the object creation are the necessary elements to run a program, whereas in Greenfoot scenarios students have to interact with objects through method calls. Results support our initial hypothesis that the dual material improves learning outcomes and suggests that practice in terms of exclusively programming might be not enough to understand abstract concepts. Comprehension of such abstractions might require special techniques like visualization.

TABLE VI. THE IMPROVEMENT PERCENTAGE FOR EACH QUESTION

	GG(Greenfoot)	**CG**
Class	46,67 %	66,67%
Object	60 %	44,44%
Instantiation	66,67 %	66,67%
Method	60 %	44,44%
Call a method	73,33 %	16,67%

With regard to the students' perception in the use of the dual instructional material, all students considered program visualization as a good mechanism to understand better the object-oriented concepts. About the usability of the Greenfoot scenarios, most of the half students perceived as easy to use Greenfoot (60% quite a lot to a lot) and easy to display the scenarios (60% quite a lot to a lot). Students (60% quite a lot to a lot) found the Greenfoot scenarios as more useful to understand the concepts of call methods and collaboration between objects.

Finally, although the scenarios seem to be useful learning resources from the obtained results that perception is not shared by students. Just 53 % considered the point scenario as useful and 40% for the geometric figure scenario. This is a typical problem in CS1 where examples do not solve any real problem so students do not perceive their utility [18]. However, complex scenarios are also difficult to visualize.

VI. CONCLUSIONS

The systematic utilization of educational tools for teaching programming is well documented in the literature. Visualization is considered as a mechanism to concretize teaching and provide students with visual feedback to reinforce their understanding.

We have studied if the use of dual instructional material (textual and visual representation of programs) could improve the understanding of the basic object oriented concepts. According to the results here shown, the use of the dual instructional material benefits learning: we found students using the visualization performed significantly better the questionnaires about object oriented concepts. Therefore, we recommend the use of this kind of materials for reinforcing.

In addition, other benefit of this kind of instructional materials is that students and teachers do not spend extra time learning a visual environment and how to program with it. Students keep learning syntax, resolving compilation errors and working with professional environments.

These results can be considered as preliminary since only one experiment was performed. However the statistical significance of the result encourages us to continue exploring this approach. In fact we are currently working on developing more scenarios and improving the visualization process in order to tolerate better the errors in the students' programs and to unify the visual and the textual environments.

ACKNOWLEDGMENT

The authors give special thanks to Sara Tena and Rosa Romero who have developed the Greenfoot scenarios.

REFERENCES

[1] C. Hu, "Rethinking of Teaching Objects-First". Education and Information Technologies 9, 3, pp. 209-218, Sep. 2004.

[2] E. Lahtinen, K. Ala-Mutka, and H. Järvinen. "A study of the difficulties of novice programmers". SIGCSE Bull. Vol. 37, 3, pp. Sep. 2005.

[3] A. Eckerdal and M. Thuné. "Novice Java programmers' conceptions of "object" and "class", and variation theory". In Proceedings of the 10th Annual SIGCSE Conference on innovation and Technology in Computer Science Education (Caparica, Portugal), ITiCSE '05. ACM, New York, NY, pp 89-93, June 27 - 29, 2005.

[4] J. Sajaniemi and C. Hu. "Teaching programming: Going beyond `objects first'". In Proceedings of the 18th Workshop of the Psychology of Programming Interest Group (University of Sussex, Sept. 7–8). University of Sussex, Brighton, U.K., pp 255–265, 2006.

[5] A. Robins, J. Rountree and N. Rountree. "Learning and teaching programming: A review and discussion". Computer Science Education vol.13 , pp. 137–172, 2003.

[6] L. Thomas, M. Ratcliffe, J. Woodbury and E. Jarman. "Learning Styles and Performance in the Introductory Programming Sequence". Proceedings of the 33rd SIGCSE Symposium,pp. 33-42, March 2002.

[7] T. L. Naps, G. Rößling, V. Almstrum, W. Dann, R. Fleischer, C. Hundhausen, et al. "Exploring the role of visualization and engagement in computer science education". SIGCSE Bulletin, 35(2), pp. 131-152, 2002.

[8] "BlueJ". [Online]. Available: http://www.bluej.org/. [Accessed: Nov. 10, 2009]

[9] "Greenfoot". [Online]. Available: http://www.greenfoot.org/. [Accessed: Nov. 10, 2009]

[10] "Alice". [Online] . Available: http://www.alice.org/. [Accessed: Nov. 10, 2009]

[11] S. Georgantaki, and S. Retalis. "Using Educational Tools for Teaching Object Oriented Design and Programming". Journal of Information Technology Impact, Vol. 7, No. 2, pp. 111-130, 2007.

[12] P. Gross and K. Powers. "Evaluating assessments of novice programming environments". In Proceedings of the First international Workshop on Computing Education Research (Seattle, WA, USA, October 01 - 02, 2005). ICER '05. ACM, New York, NY, pp. 99-110.

[13] S. Xinogalos, M. Satratzemi and V. Dagdilelis. "An introduction to object-oriented programming with a didactic microworld: objectKarel". Computers & Education, Volume 47, Issue 2, pp. 148-171, September 2006.

[14] N. Ragonis and M. Ben-Ari. "On understanding the statics and dynamics of object-oriented programs". SIGCSE Bull. 37, 1, pp. 226-230, Feb. 2005.

[15] T. Ahoniemi and E. Lahtinen. "Visualizations in Preparing for Programming Exercise Sessions". Electron. Notes Theor. Comput. Sci. 178, pp. 137-144, Jul. 2007.

[16] "jGRASP" [Online]. Available: http://www.jgrasp.org/. [Accessed: Nov. 10, 2009]

[17] New to Java Programming Center — Young Developers [Online]. Available: http://java.sun.com/new2java/learning/young_developers.jsp. [Accessed: Nov. 10, 2009]

[18] M. Buckley. "Computing as social science". Commun. ACM 52, 4, pp. 29-30, Apr. 2009.