JKAU: Sci., **Vol. 22** No. 1, pp: 59-68 (2010 A.D. / 1431 A.H.); DOI: 10.4197 / Sci. 22-1.4

Using Alice to Teach Novice Programmers OOP Concepts

Arwa A. Al-Linjawi and **Hana A. Al-Nuaim**
Computing and Information Technology College
Computer Science Department, King Abdulaziz University,
Jeddah, Saudi Arabia
arwalinjawi@yahoo.com & hnuaim@kau.edu.sa

Abstract. One of the objective of the Computer Science (CS) educational community, when teaching Object Oriented Programming (OOP), is to improve programmer productivity, and improve modelling of the real-world using objects. Some of the new techniques that allow the students to make the connection between steps in developing an algorithm for problem solving are centered on the visualization of objects and their behaviors using 3D animation environments, such as Alice. It is the purpose of this research to assess the performance of novice programmers in KAU's female CS department in Saudi Arabia, and the effectiveness of the visualization environments –Alice– in teaching inheritance as a concept of OOP.

Introduction

Object Oriented Programming has become the paradigm of choice for software development in industry as well as academia. OOP offers certain advantages over traditional programming approaches that make software easier to build, maintain, modify, and reuse. Knowledge of Object Oriented (OO) techniques will make programmers do simple programs more efficiently and effectively [1].

The central idea of OOP is to build programs using software objects. The object is an encapsulation of protected data along with all the legal operations that act on that hidden data[2]. There are several OOP languages such as Simula, Smalltalk, Modula-3, Eiffel, C++, and Java[3, 4]. The key concepts of object orientation are encapsulation, polymorphism, and inheritance. Inheritance is powerful software development technique, by which a new class is derived from an existing one, the new class inherits attributes and behavior of the pre-existing classes[5].

Instructors of introductory programming classes are faced with the challenge of helping novice programmers learn to design, build, and debug computer programs[6]. In a large class, the challenge is often overwhelming, and teachers find themselves questioning why some novice programmers struggled in the process of learning to program. The students face problems when they try to put the pieces together, identify what constructs to use and how to coordinate those constructs, and determine what went wrong when things do not work[7].

Interviews and discussions with some of the female CS department's faculty in King Abdulaziz University (KAU), who were or are currently teaching OOP courses, revealed many problems facing the faculty in teaching concepts of OOP. These problems can be summarized as follows:

- Students do not fully comprehend the OO development environments.
- Most students lack the understanding of the OOP concepts, and have difficulties in visualizing the state spaces.
- Students can't implement some of the OO features using C++ or JAVA, especially in inheritance.
- Many OO concepts must be covered in classes with a limited time.

Also, one of the major shortcomings of object-oriented programming environments is the lack of object-oriented visualization mechanisms[8]. Using 3D animations for program visualization offers computer science instructors an approach to introduce fundamental concepts to novice programmers[9].

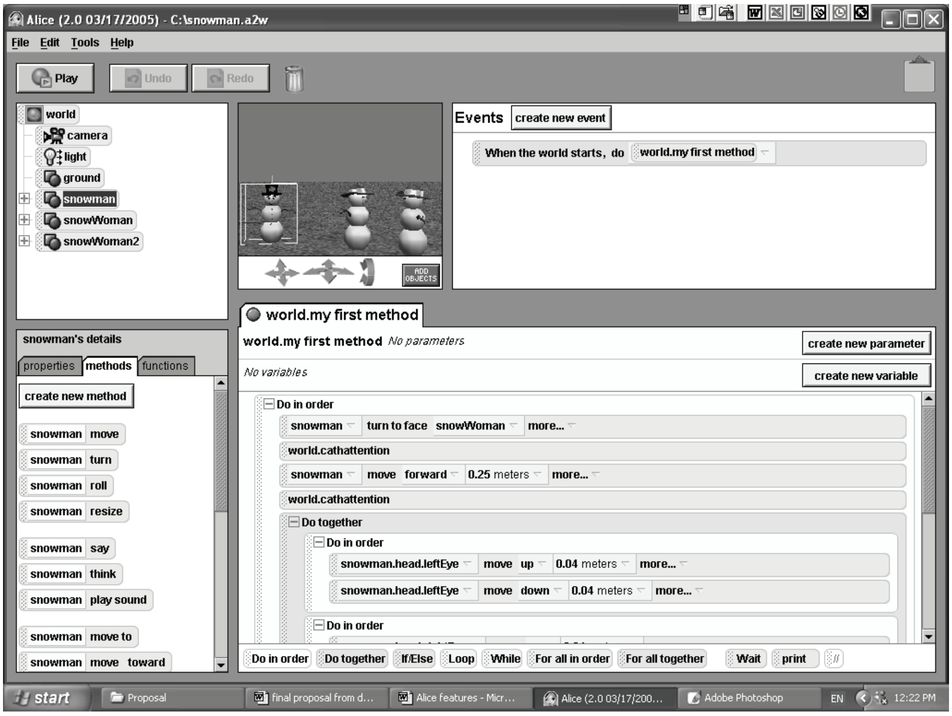
Visualization software emerged in the late 1980's for the purpose of creating and interactively exploring graphical representations of computer science concepts. Many experimental studies designed to substantiate the educational effectiveness of such visualization technology[10].

Korhonen and Malmi[11] describe a visualization system that presents novice programmers with graphical representations of algorithms which requires them to manipulate these representations in order to simulate the algorithm[10]. Teaching fundamental concepts in a first programming course using graphics and visualization in an OO environment, will provide sufficient programming experience, with stronger meaningful context for helping students to visualized object oriented concepts.

Over the last two decades, researchers have developed and used several visualization tools and technologies, such as: LOGO, GROK, KAREL, XTANGO and BALSA, Turing's World and JFLAP, TOONTALK, and ALICE.

Alice is a 3D Interactive Graphics Programming Environment built by the Stage 3 Research Group at Carnegie Mellon University under the direction of Randy Pausch[6,12]. Alice supports the pedagogical goals of teaching OO concepts, such as a fundamental introduction to objects, methods, decision statements, loops, recursion, and inheritance. A goal of the Alice project is to make it easy for novices to develop interesting 3D environments and to explore the new medium of interactive 3D graphics[12].

Alice provides an environment where students can create virtual worlds on their computers and populate them with some really interesting 3D objects in a creative sense, and use/modify these objects and write programs to generate animation. By writing simple scripts, students can control object appearance and behaviour. In fact, it is not necessary to type lines of code at all; by using the smart editor students can drag and drop the instructions that make up their programs (Fig. 1)[13].



```
world.my first method  No parameters
No variables
Do in order
    snowman turn to face snowWoman more...
    world.cathattention
    snowman move forward 0.25 meters more...
    world.cathattention
    Do together
        Do in order
            snowman.head.leftEye move up 0.04 meters more...
            snowman.head.leftEye move down 0.04 meters more...
        Do in order
```

Fig. 1. Alice interface.

Many assessment studies were focused on how the Alice's environment had an impact on a student's learning process[14]. Dann, Cooper and Pausch[9] described an approach for introducing recursion by using Alice, as part of a course in Ithaca College in New York for programmers (with no previous programming experience). They concluded that using Alice, offers computer science instructors an approach to introducing fundamental concepts to novice programmers, that allows them to quickly identify and learn from mistakes.

Rodger[15] Developed a course for non-computer science majors, at Duke University in North Carolina to teach students computer science concepts and programming. In that course students were given five visualization tools: HTML, JAWAA, StarLogo, Alice, and Karel++. Based on the evaluation survey that was given in the course, Rodger concluded that Alice was clearly the favorite and the easiest to use.

Cooper, Dann and Pausch[16] presented Alice for an objects-first strategy. They concluded that the Alice's tool was quite useful in teaching objects-first strategy to help students master the complexities of object oriented programming, it provided stronger object visualization, and a flexible meaningful context for helping students to see object oriented concepts.

Due to the nature of problems facing students while programming in an OO environment, and since the aim of Alice is to provide such an environment, which encourages the students to think in terms of objects, and represent their relationships graphically using 3D class models, the main purpose of this research was to assist the performance of novice programmers in KAU's female CS department and evaluate the effectiveness of Alice the visualization environments, in teaching inheritance as a concept of OOP, to improve their overall performance in programming.

For the purpose of this research, the treatment group was a group of students exposed to Alice during OOP lab (one lab each week for seven weeks), in addition to their traditional classes. The control group was a group of students that attended the traditional classes only . The performance of the two groups in inheritance was compared by statistically analyzing their Post test.

Methodology

1. The Sample

All undergraduate students from KAU, Computer Science Department, that had the same programming background skills, and are required to study CS203 (Programming with Java) in their traditional classes, and had no experience in using Alice tool, were asked to volunteer for the assessment lab. Twenty-four of student volunteers were accepted to attend the lab. They were taught how to implement the programming concepts that they have already covered in CS203 traditional classes using Alice without any new programming concepts. Twenty one were selected as the control group due to their similarity in their GPA to the treatment group. Both groups had two years undergraduate experience, as they had finished CS101 (Introduction to Computer) and CS102 (Introduction to C++ Programming), and had, on average, equivalent GPA's.

2. Procedure

Both groups had been tested twice. The Pre test was designed to asses the students' OOP knowledge in inheritance, even though none of the students had taken the inheritance concepts in their traditional classes. The Post-Test was designed to asses the students' inheritance knowledge after they had already taken the inheritance concept in their traditional classes, and the treatment group had attended the Alice lab.

2.1 The Treatment

The treatment was 2 hours once a week for seven weeks. During the first five weeks students were given a simple tutorial on how to use Alice's features, such as, using or creating objects and methods, using conditional statements, using functions, using parameters, creating an object, saving it to reuse it in another file, and how the new objects inherited all the attributes and methods of the old object. The students were given assignments each week that must be turned in during the lab. At the sixth week the students were asked to create a bunny family script (Fig. 2), in which every bunny inherited all the features of the parent bunny. The Bunny family's tree explains every bunny with its own methods including the methods that were inherited from the parent bunny (Fig. 3).

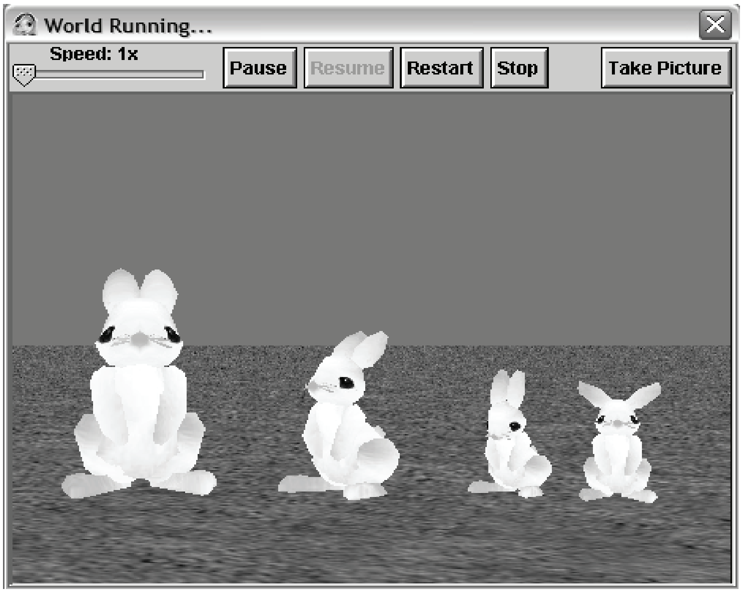


Fig. 2. The Bunny family running window.

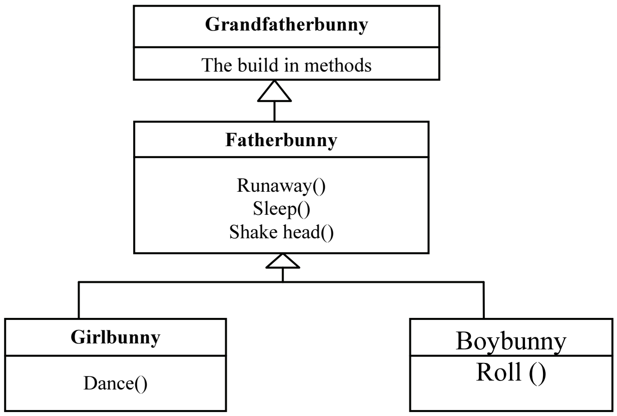


Fig. 3. The Bunny family's tree with their methods.

During the last lab (the seventh week), the students were given a computerized final quiz (the Treatment Post-test) to determine their understanding and experience in using Alice's features. The quiz covered all the concepts that were taken in the previous labs, especially inheritance.

Results

The mean results of Pre test and Post test for the treatment group were higher than the control group (Table 1). The means and the standard deviations' results in the treatment post test were divided into seven topics (Table 2). Using an object's features and using inheritance's features were the easiest and most understandable for students, illustrated by their mean scores.

Table 1. The means' results.

	Pre-test	Post-test
The Treatment Group	51.5	81.0
The Control Group	46.3	69.8

Table 2. Students' mean and standard deviation results in the treatment post-test for each topic.

Topic	Using Object's features	Using Method's features	Implementing methods	Implementing main method	Using Parameters	Using "if statement"	Using inheritance features	Total
The Mean	4.90	3.56	2.91	3.48	1.96	3.44	4.29	64.88
The Std Dev	0.37	1.11	1.02	1.11	1.71	2.21	0.87	18.93

The Independent Samples T-Test was used to determine if there was a significant difference between the results of two groups (the treatment and the control) for given variables (pre test and post test); having the alpha value in these tests assigned to 0.05 (Table 3).

Table 3. Independent samples test

		T-test	P-value(sig)
PRE	Equal variances assumed	1.087	.283
POST	Equal variances assumed	2.760	.008

When comparing the pre test and post test of both groups, the results show that both groups had the same knowledge in inheritance before they took the concept in their traditional class, and the difference between their grades (pre test) was not significant α level of 0.05. On the other hand, according to the post test results, there was a significant difference α level of 0.05 between the two groups. This means that Alice may have affected the understanding of the treatment group in the inheritance concept.

Another test was used to determine if the students' performance had been improved while using the Alice environment. A dependent sample T-Test was used to compare between the treatment group post-test's results (Written post-test) and the Alice final lab quiz's results (Treatment post-test). The Written post-test's mean was (81.0) and the Treatment post-test's mean was (64.88). Moreover, the analysis shows that this difference is statistically significant α level of 0.05 (Table 4).

Table 4. Dependent sample test for the treatment group in their written post-test and treatment post-test.

T-test	P-value(sig)
4.313	0.000

In general, students' performance was better in the written post-test than the treatment post-test to solve the problems; which may lead us to believe that the students understood the inheritance concept. However, while they were using Alice they struggled and faced some difficulties which may be due to their use of the visualization environments. Therefore a satisfaction survey was given to the students to find out what were the difficulties they faced when using the Alice environments.

This Survey aimed to develop a measure of students' experiences, as well as to better understand students' satisfaction with the use of Alice's features, as how easy the tool was , and how easy it was to implement the code.

Based on the satisfaction survey, the problems that faced the students while using Alice's features were using the parameters feature, moving a subpart of an object, and finding what they wanted easily, since some functions and methods did not give the exact meaning of their work. Using the Inheritance feature was the easiest feature according to the students' comments in the survey and to the inheritance features' mean result. Finally, many students commented in their survey that they would like to use Alice in the future because:

- Alice met and exceeded their expectations.
- Alice gave them a good visual way in learning and achieving a deeper understanding of OO concepts.
- They were able to watch what went wrong with their programs and easily debug, correct them, and learn from their mistakes.

Conclusion

During the experiment students in the Treatment group statistically performed better in OOP especially in the inheritance concept than students in the Control group. It is believed that the success of using Alice was due to the visual representation of objects. Students could see and relate to the objects and their animation actions, thus developing good intuition about objects and OOP. The researchers concluded that Alice helped students (the treatment group) to master the complexities of OOP.

The researchers' observation was that students were comfortable using objects and invoking methods on those objects. Students were able to watch what went wrong in their programs and easily debug, correct them, and learn from their mistakes.

Based on the statistical results of this research, the satisfaction survey results, and how Alice affected the students' performance in learning OOP, the researchers recommend that Alice must be integrated into the introductory OOP such as course CS203 in the CS department in KAU in order to improve a high level of students' involvement and the ability to develop an intuitive understanding of OO concepts, especially in inheritance, in a visual feedback environment.

References

[1] **Wang, P.S.** (2001) *Standard C++ with object-oriented programming* (California: Brooks/Cole).

[2] **Capretz, L.F.** (2003) A Brief History of the Object-Oriented Approach, *ACM SIGSOFT Software Engineering Notes, 28*.

[3] **Malik, D.S. (**2002**)** *C++ Programming: from problem analysis to program design* (Boston: Course Technology).

[4] **Hostetter, C.** (1998) Survey of Object Oriented Programming Languages. *Given in CS-263 (Design of Programming Languages) lecture, Berkeley University,* From: http://www.rescomp.berkeley.edu/~hossman/cs263/paper.html

[5] **Lewis, J.** and **Loftus, W.** (2005) *Java Software Solutions Foundation and Program Design* (United States of America : Pearson Education, Inc.).

[6] **Dann, W., Cooper, S.** and **Pausch, R.** (2000) Making the Connection: Programming with Animated Small World, *Proc. 5th ACM SIGCSE/SIGCUE ITiCSE Conf. on Innovation and technology in computer science education .*

[7] **Cooper, S., Dann, W.** and **Pausch, R.** (2000) ALICE: A 3-D Tool For Introductory Programming, *Journal of Computing Sciences in Colleges*, *15*.

[8] **Rosenberg, J.** and **Kölling, M.** (1997) Testing Object-Oriented Programs: Making it Simple, *Proc. 28th ACM SIGCSE technical symposium on Computer science education.*

[9] **Dann, W., Cooper, S.** and **Pausch, R.** (2001) Using Visualization to Teach Novices Recursion, *Proc. 6th ACM SIGCSE Conf. on Innovation and technology in computer science education.*

[10] **Naps, T.F., Rößling, G., Almstrum, V., Dann, W., Fleischer, R., Hundhausen, C., Korhonen, A., Malmi, L., McNally, M., Rodger, S.** and **Velázquez-Iturbide, J.A.** (2002) Exploring the Role of Visualization and Engagement in Computer Science Education, *Working group reports from ACM SIGCSE/ITiCSE on Innovation and technology in computer science education.*

[11] **Korhonen, A.** and **Malmi, L.** (2000) Algorithm Simulation with Automatic Assessment, *Proc. 5th ACM SIGCSE/SIGCUE Conference on Innovation and Technology in Computer Science Education.*

[12] **Cooper, S., Dann, W.** and **Pausch, R.** (2003) Using Animated 3D Graphics To Prepare Novices for CS1, *Computer Science Education, Routledge, part of the Taylor & Francis Group,* **13**, 3-30.

[13] **Cooper, S., Dann, W.** and **Pausch, R.** (2005) *Learning to Program with ALICE* (United state of America: Pearson Prentice Hall).

[14] **Gross, P.** and **Powers, K.** (2005) Evaluating Assessments of Novice Programming Environments. *ACM Press, Proc. of the 2005 international workshop on Computing education research,* 99-110.

[15] **Rodger, S.H.** (2002) Introducing Computer Science through Animation and Virtual Worlds, *Proc. 33rd ACM SIGCSE technical symposium on Computer science education.*

[16] **Cooper, S., Dann, W.** and **Pausch, R.** (2003) Teaching Objects-first In Introductory Computer Science, *Proc. 34th ACM SIGCSE technical symposium on Computer science education.*

استخدام الـــس (ALICE) لتدريس حديثي البرمجة مفهوم (OOP)

أروى عبدالعزيز اللنجاوي، وهناء عبدالله النعيم

قسم الحاسبات وتقنية المعلومات - كلية الحاسبات وتقنية المعلومات،

جامعة الملك عبدالعزيز - جدة - المملكة العربية السعودية

المستخلص. إن هدف المجتمع التعليمي لعلوم الحاسبات عند تعليمهم برمجة الـــ (OO) هو السماح بتطوير إنتاجية المبرمجين، رفع مستوى المرونة والقوة، تحسين للهيكلية كصورة واقعية باستخدام الـــ (Object)، وزيادة إعادة استخدام شفرات البرمجة. إن ربط حلقة الوصل بين كتابة خطوات الـــ (Algorithm) وإنشاء خاصية البرمجة، تعتمد إلى حد كبير على خبرة الطالب في هيئة وبيئة البرنامج، وكيفية تغير الحالة عندما يتم تنفيذ البرنامج المنشأ. بعض التقنيات الجديدة التي تسمح للطالب بربط حلقة الوصل بين كتابة خطوات الـــ (Algorithm) وإنشاء خاصية البرمجة متمركزة على رؤية الـــ (Objects) وسلوكياتها باستخدام بيئة رسومات حركية ثلاثية الأبعاد، مثل ألس (ALICE). إن الغرض من هذه الرسالة التعليمية هو تقييم أداء حديثي البرمجة في جامعة الملك عبدالعزيز، شطر الطالبات، قسم علوم حاسبات، وفعالية بيئة الرؤية كألس (ALICE) في تعليم مفاهيم برمجة الـــ OO بمثل مفهوم الوراثة.