

# Teaching Java with Robots and Artificial Life

R. Mark Leeman and David H. Glass  
School of Computing and Mathematics  
University of Ulster  
Newtownabbey, Co. Antrim, BT37 0QB, UK  
Email: [dh.glass@ulster.ac.uk](mailto:dh.glass@ulster.ac.uk)

## Abstract

This paper investigates the use of simulated robots and an artificial life simulation in the teaching of Java to first year students. The views of first year students were sought on how these environments compared with the more traditional approach that had been used previously. The results show that significant advantages could arise from using such approaches, particularly in terms of enhancing student interest in programming.

**Keywords:** teaching Java, robots, artificial life, simulation, object oriented programming

## 1 Introduction

It is generally acknowledged among students and lecturers that programming is a difficult subject. Many students struggle with programming at an introductory level and this in turn gives rise to problems later on in their degree course. While there has been much discussion about the most appropriate language to use for introductory programming, the problems seem to be of a more general nature and so lecturers are often unsure how to address the issue. Research has emphasised difficulties due to the intrinsic nature of programming itself as well as problems due to lack of motivation on the part of students (see, for example, Jenkins 2001, 2002).

Marton and Säljö (1976) identify two styles of learning, 'surface' where the focus is on memorising facts and 'deep' where the learner focuses on gaining an understanding of the subject. Jenkins (2002) points out that surface learning can be useful for learning syntax, but that deeper understanding is required for other aspects of programming. In particular, the process of designing algorithms appears to require deep learning. Coupled with the fact that programming can be seen as a set of hierarchical skills, many of which may need to be used at a given time (Jenkins, 2002), it is clear that students need to be well-motivated to succeed at programming. But this raises the question as to what type of motivation is required and whether anything can be done to enhance it.

A great deal of attention has been given to the topic of motivation in educational research (see for example Brown et al, 1998). Entwistle (1998) outlines three types of motivation: extrinsic motivation, which focuses on satisfactory completion of the course; intrinsic motivation, which derives from interest in the subject area; and achievement motivation, which focuses on personal level of achievement. He also points to a link between these types of motivation and learning approaches, with extrinsic motivation leading to a surface approach to learning and intrinsic motivation leading to a deep approach. Consistent with the importance of deep learning for programming as noted above, is the study of failing students by Sheard and Hagan (1998) who found that such students are likely to have a primarily extrinsic interest. Unfortunately, the study of Jenkins (2001) found little evidence of many students with

an intrinsic interest in programming. Thus, addressing this lack of intrinsic interest seems to be an important goal for programming education.

One aspect of this problem is the way in which teaching is carried out especially in light of the reputation of programming as a difficult subject (Jenkins, 2002). Pendergast (2006), for example, emphasises that a good introduction to programming module should leave students with a positive impression of programming as well as promoting good programming habits and techniques, while Guo (2006) points to the benefits of visual material and learning aids in teaching. Another proposal is to move away from using traditional unimaginative programs that, for example, simply print out the highest mark achieved by a group of students in an exam, to more interesting topics such as games that promote the logical and critical thinking required to translate an algorithm into a program (Rajaravivarma, 2005). This focus on programming rather than simply learning a set of techniques is also promoted by Gries (2006) who explains that a programming class should include much more than simply showing a student a program and asking them to re-write it using the technique covered in the previous lecture.

The goal of enhancing students' interest in programming through the use of simulated robots and an artificial life simulation is the primary focus of this paper. It is also anticipated that other aspects of learning, such as understanding of object orientation, may benefit from the visual nature of the programs. The environment used is that created by Becker (2001, 2007) which is essentially a Java implementation of Karel the Robot, initially developed by Pattis (1981) in C++. The artificial life simulation uses a predator-prey model (see, for example, Flake, 2000). While artificial life has been used for teaching C++ (Pattis, 1997), we are not aware of any research on its influence on student interest or its effectiveness in teaching Java.

The evaluation of our approach was carried out by assigning exercises to a group of first year students taking their first object oriented programming module, although they had previously completed a module in C. A questionnaire was used to compare the robot environment of Becker and the artificial life simulation with the more traditional approach currently used in their module. Informal discussions were also

held with the same group of students and with a small group of final year students and recent graduates to obtain their views on the new approaches.

The rest of the paper is set out as follows. Section 2 describes the simulated robot environment and its extension to the artificial life simulation. Section 3 discusses the feedback received from students and section 4 presents some conclusions.

## **2 Robots and Artificial Life**

As noted earlier, the environment created by Becker (2007) for simulated robots was used in this research. This occurred in two ways: first, the standard environment was used and several exercises, similar to those found in the textbook, were set to students; second, it was extended to construct an artificial life simulation which was then used as the basis for setting exercises to students. In this section we provide an overview of the environment and then present the artificial life simulation.

### **2.1 The Simulated Robot Environment**

The simulated robot environment consists of a simple interface in which robots, normally represented by red triangles, move about either horizontally or vertically on a grid. It was specifically designed to teach programming in Java to first year students. The accompanying textbook uses robots to guide students through the basics of programming before moving onto more complex topics which in many cases do not involve robots. Basically, the robots package is a set of classes defining an environment in which students can create objects on a grid and call associated methods to determine their behaviour. Robots are created as objects in this way and can be instructed to move, to turn, to pick things up, etc. Other types of objects can be created such as walls to block robots and streetlights that can be turned on and off. Students can write new classes so that new types of objects can be added.

Becker (2007) points out a number of advantages to this approach in terms of visualisation, ease of programming (because it is easy for students to imagine carrying out tasks such as moving, picking things up, etc.), students having fun programming robots and introduction of objects from the outset. Becker also emphasises the pedagogical advantages of using his approach since it enables

students to first use robots and only after that to write new classes defining new kinds of objects.

One limitation of the environment is that robots may only move in one of four directions (North, South, East and West) and so lack the mobility that is present in other environments. This feature also has a positive side to it, however, since it makes it easier to specify the direction to move in. A further restriction is that the environment itself is closed source. This means that while various classes can be extended to allow the functionality required for the artificial life simulation, the original methods cannot be modified and so need to be overridden.

## **2.2 The Artificial Life Simulation**

Artificial life has two goals: to use computer systems to describe the qualities of living systems and to attempt to artificially create a living creature within a computer medium (Sullins, 2005; Pollack et al, 2004). Emmeche (1994) identifies the ability of living things to autonomously decide where and when they will move as a defining characteristic. He also proposes that an artificial life form should be a self-contained entity which endeavours to maintain itself. Irrespective of how the relationship between artificial life forms and living creatures is to be understood, it is clear that the former must be able to simulate, at least to some extent, some of the characteristics of the latter. Here we present a brief overview of the artificial life simulation that was implemented in the Becker environment.

The artificial life simulation in this work was a predator-prey model for which the rules were initially based on Flake (2000) and were built upon the idea that each robot has a finite supply of energy and so it must be replenished periodically by eating something lower down on the food chain than itself. The amount of energy a robot has at any moment will determine how it will act. If a robot has a large amount of energy remaining it will have no need to eat and so will attempt to perform another task such as finding a mate to produce offspring, while a robot with very little energy left will focus solely on finding something to eat. Each robot also has a metabolism: prey have a low metabolism and so will need to eat less frequently than predators. The result of this is that prey will spend more time reproducing, thus maintaining a sufficient amount of food to maintain the populations of predators. Prey will always

run from a predator regardless of whether the predator needs to eat or not. All of the predefined variables required for these rules can be modified by users to customize the simulation and to investigate the effect different values will have on the outcome of the simulation.

The main classes for the simulation are a *Village* class and *EnhanceRobot* class which extend the *City* and *RobotSE* classes respectively of the original environment. There is also a *Grass* class which extends the original *Thing* class and provides the source of food for the prey. The main method creates an instance of the *Village* class and then creates an array of instances of the *EnhancedRobot* class, each running in a separate thread. The *Village* object is passed to the *EnhancedRobot* objects as one of their parameters. The array of *EnhancedRobots* contains both predators and prey depending on their attributes. The most critical aspect of the system was to allow robots to interact, a process not included in the original environment, so that predators could kill prey, reproduce, etc. This was achieved by effectively using the *Village* class as an intermediary to allow each individual robot to directly interact with every other robot.

### **3 Evaluation**

To evaluate the original Becker environment and the artificial life simulation as teaching aids, a practical class was arranged for a group of first year students to try out a series of exercises. Feedback was obtained via informal discussions and a questionnaire. The second form of feedback was obtained through informal one-to-one interviews with a small group of final year students and recent graduates after they had tested the system.

#### **3.1 First Year Students**

The exercises for the first year students were divided into two sections. The first section focused on the original Becker environment without any modifications and used exercises either from the accompanying book or based upon them. These included an exercise which involved moving a robot to a desired location by calling various methods and writing new methods to add additional functionality to robots. The second section focused on the artificial life simulation with several exercises

being drawn up requiring the modification of existing code to monitor the numbers of predators and prey as the simulation runs and to alter the robots' behaviour.

During the session it was apparent that students were taking a greater interest in the exercises than would normally have been the case with the more traditional exercises that had been used previously. Whenever they encountered a problem with their code they were usually able to identify the nature of the problem quite easily since it was apparent from the visualisation. The students were also more eager to ask questions to the instructors, whose task was made somewhat easier due to the fact that students were better able to identify and explain the problem than in previous practical classes.

In the questionnaire students were asked to give their opinions on the Becker environment and the artificial life simulation as well as on the more traditional approach used in practical classes earlier in the module. When asked to identify any advantages and disadvantages of the different approaches a large number of students (81%) stated that the exercises using the Becker environment and the artificial life simulation were more interesting than the more traditional exercises. The visual nature of the new approaches was also identified as a benefit by many students (52%). The only disadvantage of the new approaches identified by more than one student was that they were not able to see all the code for the system (14%). No significant advantages of the more traditional exercises emerged. The only notable difference between the Becker environment and the artificial life simulation emerging from the comments was that several students found the latter more challenging.

To elicit a summary of their overall attitudes to the new approaches compared with the traditional approach, students were asked to rate first of all the Becker environment against the traditional approach and then the artificial life simulation against the traditional approach. In both cases a scale from -5 to +5 was used, with -5 indicating a strong preference for the traditional approach, 0 indicating indifference and +5 indicating a strong preference for the Becker and artificial life environments respectively. The results are presented in table 1. In both cases the result is found to be significantly different from 0,  $p < 0.001$ , using a t-test. Since an average value of 0

would correspond to indifference these results clearly point to a preference for the new environments over the traditional approach. Furthermore, there is no statistical difference between the mean values obtained in the two cases ( $p = 0.913$ ).

Table 1. Average ratings for the Becker environment and artificial life simulation compared with the traditional approach. Ratings ranged from -5 (Strong Preference for Traditional Approach) to +5 (Strong Preference for Becker / Artificial Life).

	Mean Rating
Becker environment vs. Traditional approach	2.71
Artificial life simulation vs. Traditional approach	2.76

An informal discussion was also held with the cohort several weeks after the practical class had taken place. The attitudes towards the new environments were consistent with those discussed above, with two topics in particular coming to the fore. First, students found the new environments much more interesting than the more traditional approach used in the module and as a result they were better motivated to complete the exercises and to explore the environment further outside the allocated class time. Second, students found the visual nature of the new environments helped them to gain a better understanding of programming concepts such as object orientation. The general opinion of the cohort was that they would recommend the Becker environment to be used for future first year classes. They did, however, find the artificial life simulations more demanding and thought they could be used towards the end of the first year module or in second year modules.

### 3.2 Final Year Students / Graduates

While the main focus of the evaluation was to gather the attitudes of first year students, discussions were also held with a group of four final year students and recent graduates from computing subjects to determine what they thought the advantages and disadvantages of using the original Becker environment and the artificial life simulation were, compared to their own experiences of programming modules.

A point that recurred throughout the discussion with the final year students was the obvious nature of objects within the Becker environment. Rather than a vague

concept only half remembered from the previous lecture, this system made objects instantly recognisable to the user and it was felt that such a graphical representation would have been beneficial, particularly in the early stages, to their learning of object oriented programming. The discussions also focused on the exercises that could be based on this system and it was felt that these would be much more interesting than the sometimes very 'clinical' exercises from their own experiences.

Despite the view that basing exercises on an artificial life simulation would make them more interesting, it was felt that this may confuse or even scare students when they are suddenly presented with, and asked to modify, long class files that use methods and techniques that they will not cover for at least another year. It was also felt that it would be more beneficial for students to work on a program from scratch, building it up over the course of the module, rather than modifying someone else's code.

There were also some concerns about the Becker environment becoming a 'kid gloves' approach that, while initially helping students, may cause them more difficulties once they progress on to creating 'real' programs that are required to run independently of any other software.

## **4 Conclusions**

This paper has investigated the use of the simulated robot environment created by Becker (2007) and an extended artificial life simulation using the same environment in teaching of Java programming to first year students. This was done by setting a series of programming exercises in each of these environments to students who were taking a first year object oriented programming module and, with the exception of these exercises, had been set more traditional exercises in previous practical sessions. Informal discussions were also held with final year students and recent graduates.

While the results are preliminary in nature, they indicate the benefits of both the original Becker environment and the artificial life simulations in terms of increased student interest and their perceived enhancement of understanding concepts in

object oriented programming due to their visual nature. While comments from students did seem to indicate some preference for the Becker environment over the artificial life simulation, there is no reason why both could not be used with the artificial life simulation being used for later parts of a first year module and for setting some exercises which would stretch stronger students.

While the primary focus in this paper has been on enhancing students' *interest* in programming, we would of course really like to see that such environments would also enhance their *learning*. Nevertheless, as discussed in the introduction, lack of motivation, particularly intrinsic motivation, appears to be a significant factor in poor learning and so it is hoped that enhancing the interest would also enhance learning. This is clearly an area for further work and, given the results of this study, it is one we plan to consider in the future.

Other directions for future work could involve further development of the artificial life simulation and the use of such environments in more advanced programming classes. The artificial life simulation might be particularly important in this latter respect since it involves some components which go beyond what is usually covered in a first year module.

## **Acknowledgements**

The authors would like to thank Alexander Grigorash and anonymous referees for helpful comments.

## References

- Becker, B.W. (2001) Teaching CS1 with Karel the Robot in Java, *ACM SIGCSE Bulletin*, 33 (1), pp. 50-54
- Becker, B.W. (2007) *Java: Learning to Program with Robots*, Thomson, Boston, MA.
- Brown, S., Armstrong, S. and Thompson, G., (eds.) (1998) *Motivating Students*, Kogan Page, London.
- Emmeche, C. (1994) *The Garden in the Machine: The Emerging Science of Artificial Life*, Princeton University Press, Princeton
- Entwistle, N. (1998) Motivation and Approaches to Learning: Motivating and Conceptions of Teaching, In Brown et al (1998)
- Flake, G.W. (2000). *The Computational Beauty of Nature*, MIT Press, Cambridge, MA.
- Gries, D. (2006) What have we not Learned about Teaching Programming? *Computer*, 39 (10), pp.81-82
- Guo, H. (2006) Visual Material and Learning Aids in Teaching Object Oriented Programming, In *Proceedings of the 7th Annual ICS HE Academy Conference, Dublin, Ireland (2006)*, pp. 64-69
- Jenkins, T. (2001) The Motivation of Students of Programming, In *Proceedings of ITiCSE, Canterbury, UK (2001)*, pp.53-56
- Jenkins, T. (2002) On the Difficulty of Learning to Program, In *Proceedings of the 3rd Annual LTSN-ICS Conference, Loughborough, UK (2002)*, pp.53-58
- Marion, F. and Säljö, R. (1976) On Qualitative Differences in Learning I: Outcome and Process, *British Journal of Educational Psychology*, 46, pp.4-11
- Pattis, R.E. (1981) *Karel the Robot: A Gentle Introduction to the Art of Programming*, John Wiley and Sons, New York
- Pattis, R.E. (1997) Teaching OOP in C++ using an Artificial Life Framework, *ACM SIGCSE Bulletin*, 29 (1), pp.39-43
- Pendergast, M.O. (2006) Teaching Introductory Programming to IS Students. *Journal of Information Technology Education*, 5, pp.491-515
- Pollack, J., Bedau, M.A., Husbands, P., Ikegami, T and Watson, R.A. (eds.) (2004) *Artificial Life IX: Proceedings of the Ninth International Conference on the Simulation and Synthesis of Living Systems*, MIT Press, Cambridge, MA.

- Rajaravivarma, R., (2005) A Games-Based Approach for Teaching the Introductory Programming Course, *ACM SIGCSE Bulletin*, 37 (4), pp.98-102
- Sheard, J. and Hagan, D. (1998) Our Failing Students: A Study of a Repeat Group, In *Proceedings of ITiCSE, Dublin, Ireland (1998)*, pp. 223-227
- Sullins, J.P. (2005) Ethics and artificial life: From modeling to moral agents. *Ethics and Information Technology*, 7, pp. 139-148