Problems First, Second, and Third

Gary Hill, University of Northampton, Northampton, UK
Scott Turner, University of Northampton, Northampton, UK

ABSTRACT

This paper considers the need to focus initial programming education on problem-solving, prior to the teaching of programming syntax and software design methodology. The main vehicle for this approach is simple Lego based robots programmed in Java, followed by the programming of a graphical representation/simulation to develop programming skills. Problem solving is not trivial (Beaumont & Fox, 2003) and is an important skill, central to computing and engineering. The paper extends the authors earlier research on problems first and problem solving (Hill & Turner, 2011) to further emphasise the importance of problem-solving, problem based learning and the benefits of both physical and visual solutions. An approach will be considered, illustrated with a series of problem-solving tasks that increase in complexity at each stage and give the students practice in attempting problem-solving approaches, as well as assisting them to learn from their mistakes. Some of the problems include ambiguities or are purposely ill-defined, to enable the student to resolve these as part of the process. The benefits to students will be discussed including students’ statements that this approach, using robots, provides a method to visually and physically see the outcome of a problem. In addition, students report that the method improves their satisfaction with the course. The importance of linking the problem-solving robot activity and the programming assignment, whilst maintaining the visual nature of the problem, will be discussed, together with the comparison of this work with similar work reported by other authors relating to teaching programming using robots (Williams, 2003). In addition, limitations will be discussed relating to the access to the physical robots and the alternative attempts to simulate the robots using three options of, Microsoft Robotics Studio (MSRS), Lego Mindstorms and Greenfoot simulators.

Keywords: Computational Thinking, Graphical Programming, Graphical User Interfaces, Problem Based Learning, Problem-Solving, Project Based Learning, Robots, Simulation

INTRODUCTION

This paper considers the ‘evolution’, ‘realisation’, ‘optimisation’ and ultimately the ‘evaluation’ that for undergraduate first year programming ‘problem solving should be taught prior to a programming language’. The hypothesis is that ‘problem solving should be taught as a fundamental skill and then the programming language can be used/taught as a tool for solving problems’, originated during discussion at a Java and the Internet in the Computing Curriculum.

DOI: 10.4018/IJQAETE.2014100103
This paper, predominantly discusses the teaching of programming and problem solving to undergraduate first year computing students, using robots and graphical programming to emulate the robot tasks. It also introduces alternative opportunities to use robot simulators, where access to physical robots is limited.

Probably the most important skills a computer scientist or engineer must possess are those of problem solving, to the point where the role of computers as problem-solving tools is seen as increasingly useful (Gallopoulos et al, 1994). These skills are highlighted in numerous benchmark and guideline statements for engineering and computing (Houghton, 2004; Adams et al., 2008; Adams and Turner, 2008). While it is appreciated that being a good problem solver involves knowledge and experience, there are other interventions, such as training and practice that can improve process skills in engineering and computing undergraduates.

Mindstorm based robots have been used previously for teaching programming to computing and engineering students (Fagin, 2003; Lawhead et al., 2003; Price et al., 2003; Williams, 2003). Here we make use of them for problem solving. Synergies can be achieved using robots to develop problem-solving skills and skills of pre-object programming (for example Culwin et al., 2006) and simulation of robots for teaching programming as a visual approach using the widely used programming language Java.

The approach discussed here focuses upon the development of problem-solving skills first and not on learning a new programming language from the outset. Therefore, initially, any programming is kept simple with the minimum of commands, with objects unknowingly used, as these are later introduced/learned during the programming stage of the computing module. Work within the team (Turner & Hill, 2008) suggests that using Lego robots within the teaching of problem-solving and the resulting java graphical user interface (GUI) emulation has some benefits for the students. The students come on the courses with a range of experiences and abilities, but many of them have limited or no experience of programming (Jenkins and Davy, 2002). In other words, this is likely to be the first time they are exposed to programming or expected to program. It was felt within the course team that giving the students a tool that is non-threatening and that would also provide early success in a physical and visual form (which, if it were to break, it would not matter) would be beneficial. Lego-based robots are initially used, followed by the same problems being repeated as a graphical representation/simulation to satisfy these same criteria.

Other authors have noted the suitability of robots for teaching programming. Lawhead et al. (2003) stated that robots provide entry level programming students with a physical model to visually demonstrate concepts, that the most important benefits of using robots in teaching introductory courses is the focus provided on learning language-independent truths about programming and programming techniques and that robots readily illustrate the idea of computation as interaction. Robots have also been highlighted as a tool for improving engagement with group work (Gold, 2010). This paper advances the idea that developing problem-solving skills first, then embarking on programming concepts, constructs and principles, is beneficial. We explore this idea in detail by discussing the module where these ideas have been tested.

The module (Table 1) is structured into two parts, eight weeks (16 hours) spent on problem-solving, followed by sixteen weeks (32 hours) of graphical programming in Java. The underlying approach is that as the module develops the focus evolves from general concepts of problem-solving (e.g. brain-storming, functional decomposition) to solving problems based around robots, which increase in difficulty (but not necessarily in complexity). The module is first assessed by a robot-based project (50%), which then leads into developing the same problem via a graphical user interface in Java, which is finally assessed (remaining 50% of module assessment).
Design of Assessment Information System for Program Accreditation
www.igi-global.com/article/design-of-assessment-information-system-for-program-accreditation/163288?camid=4v1a

Teaching Software Architecture in Industrial and Academic Contexts: Similarities and Differences
www.igi-global.com/chapter/teaching-software-architecture-in-industrial-and-academic-contexts/102342?camid=4v1a