Common Environment for Undergraduate Computer Programming

Colin Price
University College Worcester
Worcester WR2 6AJ
United Kingdom
44-1905-855316
c.price@worc.ac.uk

John Price
University College Worcester
Worcester WR2 6AJ
United Kingdom
44-1905-8555060
j.price@worc.ac.uk

ABSTRACT

We present an Integrated Environment suitable for learning and teaching computer programming which is designed for both students of specialised Computer Science courses, and also non-specialist students such as those following Liberal Arts. The environment is rich enough to allow exploration of concepts from robotics, artificial intelligence, social science, and philosophy as well as the specialist areas of operating systems and the various computer programming paradigms.

Categories and Subject Descriptors
K.3.2 [**Computers and Education**]: Computer and Information Science Education.
General Terms: Design, Experimentation.

1. INTRODUCTION

Various environments are usually employed with computer science students to support learning of core elements such as Operating Systems, Artificial Intelligence and Programming. Often the environment used to support each of these areas is distinct. Students may be required to program in Lisp, Java or C and work with many Development Environments all within a short period of time. This entails spending a lot of time learning application details which could be better given to fundamental concepts.

We are have developed a unified learning environment, a combination of hardware and software which supports learning and teaching in Robotics, Artificial Intelligence, Programming, and Operating Systems. The environment is based upon the programming of autonomous robots (in-house developed Lego robots in "C") and Java multirobot simulations (also using "C"-code). The robot electronics uses an industry-standard 8051 microcontroller derivative. Students combine the controller board with Lego components and a selection of sensors. The robots are programmed in 'C' using an industry standard IDE, supplied by Keil. The emphasis has been to devise a minimalist robot which can satisfy many educational goals with a premium cost while using industry-standard components. Students find the use of industry-standard components highly motivating, these systems are "real-world" and learned skills are transferrable. The dual use of physical robots and Java simulation is a second keystone of our architecture.

The programming tasks are behaviour-based and include FSMs, subsumption, Arkin's schemas and neural net paradigms.

We have written a tiny operating system that supports multitasking within 4kB of microcontroller RAM. Each behaviour runs as a separate thread within the operating system.

Working with these small robots, students individually learn concepts of programming, multi-tasking operating systems and behaviour-based control. They also come together as teams and investigate interactions and cooperation between multiple autonomous robots. Students develop team (and project management) skills.

The simulation environment for these robots also supports single or multiple entities. This is written as a Java application, but allows the students to use almost identical C-code in the Java application to program the simulated robot. This reduces the complexity for courses where programming is not the main consideration. The simulation runs each individual robot's behaviour as a separate Java thread, each entire robot is run as a thread group. The current simulation also supports direct programming in Java and a table-based Finite State Machine language. The underlying use of Java follows from our adoption of the Stein's concurrent process view of computation, as well as being the modern Web language.

One key objective of our development was to facilitate Operating Systems courses where learning traditionally is either done by limited experimentation with a real OS such as UNIX, or else via simulations such as MOSS. Both have their limitations. The embedded RTOS we have developed provides a richer learning experience, allowing exposure to most operating system concepts in a live system. Student experimentation is highly rewarding, since the behaviour of the robots immediately reflects the changes made by the students to the OS structure or parameters.

This research project has been running for almost one year. The experience of first year undergraduates is now being used to refine the learning approaches as well as system hardware and software. We have also carried out (and continue to conduct) trials with school children of ages 13-18. The educational objectives of this project are simple. We believe in a constructivist educational paradigm, but one which is grounded or embodied in the real world. Simulations must coexist with physical robots. Students observe robot behaviour and directly observe the effects of their programming experiments. They are learning to be creative via synthesising behaviour, and their hypothesis-testing skills are being exercised in this problem-based environment. Moreover, through programming behaviour, they are coming into contact with concepts in AI, cognitive psychology and have shown the ability to question the nature of the Self.