Hands-on Artificial Intelligence Education Using LEGO Mindstorms™: Lessons Learned

J. Ben Schafer
Department of Computer Science
University of Northern Iowa
schafer@uni.edu

Abstract

Educational researchers have suggested that instruction utilizing a variety of delivery modes helps students with differing learning styles to better understand the studied material. While introductory CS courses frequently contain "hands on" application of the material, many upper division courses seem to focus largely on the concept of "reflective practice."

This paper will focus on the author's attempts to provide a series of hands on activities conducted as part of an Artificial Intelligence course. "RoboLab" was an optional, 1-credit lab offered in conjunction with the more traditional 3-credit AI course. This lab, using the LEGO Mindstorms™ robotics platform and the leJOS firmware, allowed students to apply a variety of the AI topics studied in the classroom to the construction/creation of student-built robots to solve a diverse set of tasks. These AI topics included problem solving, as well as rudimentary knowledge acquisition and planning, and multi-agent communication.

Introduction

Educational researchers have long suggested that instruction utilizing a variety of delivery modes helps students with differing learning styles to better understand the studied material. Few, if any of us, would think about teaching our introductory programming courses solely via textbook readings and lecture. We firmly believe that, to learn, students must "do." Thus, we provide a variety of assignments and "labs" that allow students to participate in both reflective practice in the classroom, and active practice in the computer lab. Thomas et al. [7] have studied the success rate in traditional CS I courses by students exhibiting each of the different learning styles described in the work of Richard Felder [2] and suggested that instruction targeted across styles greatly improves the performance of students from all styles, but in particular from those styles less prominent in the domain of computer science.

Despite all of this, many of us seem to push aside this belief when we teach our upper division courses. It is not uncommon to find these courses being taught with little or no "hands on" application of the techniques being studied. Is this because we suddenly feel these principles become less important as students mature, or are we are simply modeling what we know ("Our instructors didn't provided hands on instruction. Why should we?")? If it is the latter, than perhaps it is time we examine these upper-division classes and consider where a more active learning approach may be appropriate.

This paper describes the author's attempts to provide a series of hands on activities conducted as part of the Artificial Intelligence course taught during the fall semester of 2003 at the University of Northern Iowa. Modeled, in part, on an AI course taught by Frank Klassner at Villanova [1], "RoboLab" was an optional, 1-credit lab offered in conjunction with the more traditional 3-credit AI course. This lab, using the LEGO Mindstorms™ robotics platform and the leJOS firmware (a Java based OS), allowed students to apply a variety of the AI topics studied in the classroom to the construction/creation of student-built robots to solve a diverse set of tasks. These AI topics included problem solving – using simple search, informed search, and exploration techniques such as hill climbing and simulated annealing – as well as rudimentary knowledge acquisition and planning, and multi-agent communication.

Organization of the Lab

In an effort to provide a hands-on environment for learning and discussing artificial intelligence, the author spent the spring of 2003 (and approximately $3000) planning and constructing an eight station "RoboLab" to be used as an optional supplement to the artificial intelligence course offered at the University of Northern Iowa. This section will discuss the equipment used in RoboLab, including the hardware, site needs, and software, as well as the course structure for such a lab.

Equipment

While there are a variety of "cost accessible" platforms for building robots at the undergraduate level, including HandyBoard and perhaps even ActiveMedia robots, we decided to use the LEGO Mindstorms platform. This decision was made for a variety of reasons including overall cost, ease with which students can construct and modify a variety of robots, and the "fun factor" – most of the students participating in RoboLab chose to do so largely because they remembered how much fun it was to play with LEGOs as a child.

The LEGO equipment was purchased directly from LEGO's website. While, historically, LEGO has chosen to not offer any form of educational or bulk discounts they do routinely offer a "Robotics Invention System Kit" (RISK) via their print and online catalogs. This kit consists of the standard Robotics Invention System (RIS™) bundled with additional sensors and pieces. The actual contents of this kit changes over time depending on what they choose to promote (or more pessimistically, what they chose to clear out of their warehouse). For example, at the time that we chose to purchase our equipment, the System Kit came with the RIS 2.0, the Ultimate Builder's Challenge, a remote control, as well as additional motors and a capacitor. In addition to the RISK, we chose to purchase a rotation sensor for each team. The total purchase price was approximately $250 per team.

In addition to the robotics equipment, each team was issued eight rechargeable AAA batteries (the "computer" portion of the robot, the RCX™ brick, requires six batteries) and a charger able to handle all eight batteries at once. The total purchase price was approximately $45 per team.

At this point, we had invested over $2400 in equipment. Clearly, it was essential to have a means by which student teams could organize and securely store the equipment that they were issued. To achieve this, we purchased a multi-tray, parts-organizer and a 2-gallon Rubbermaid storage bucket for each group. Small parts were sorted by type, size, and color into the parts organizer, while larger or oddly shaped parts were sorted into Ziploc storage bags and kept in the storage bucket. The bucket also doubled as a container for their work in progress. Finally, we purchased two Rubbermaid locker units. Each unit features four independently lockable storage spaces approximately 18x18x36. In addition to plenty of room for both parts containers, teams had room for textbooks and other materials they were working with. The total purchase price of lockers and storage units was approximately $55 per team.

In addition to a secure and organized work environment, we needed a lab setting where students could both work on their programs and run their robots around the given interaction environment with as little disturbance as possible. Fortunately, we were able to obtain the shared use of a teaching lab. This lab contained six networked computers specifically for the use of RoboLab participants, a large "testing table" (an old conference room table which would comfortably support a 4x8 foot sheet of plywood, as well as a variety of "work" tables and chairs that could be rearranged as needed.

LEGO distributes the Mindstorms with a powerful visual programming environment often referred to as ROBOLAB. This environment is a great example of how to write a "clean," intuitive interface. However, it is targeted towards an audience half the age of the typical undergraduate. While an intelligent seven-year-old could use it to develop fairly sophisticated programs, the visual, drag and drop, environment is probably too simplified for the typical AI student, and there are many activities that would be fairly difficult if not impossible to perform using this base environment.

Fortunately, several free, third-party, firmwares (operating systems) exist for the Mindstorms platform. These include NQC (not quite C) [1], leJOS (LEGO Java Operating System) [4], and RCXLisp [5]. Since the students at UNI receive a minimum of two semesters of programming instruction in Java, we chose to use the leJOS firmware for the labs. Students write code in a slightly modified version of Java, compile using the leJOS compiler, and then download the byte code to the RCX brick. leJOS is open source, readily available online [4], and several helpful tutorials exist for installing and programming using this environment.

Course Structure

The artificial intelligence course at UNI has traditionally been available in either a three-credit or four-credit version. The three-credit version was available for MIS majors from the College of Business and involved no actual programming. The four-credit version was required for departmental majors and consisted of additional programming activities. Recent restructuring of the CS majors has left non-majors largely ineligible for this course due to pre-requisite issues, yet both sections were still on the scheduling books.

Taking advantage of this, it was decided that the offering of AI presented in the fall of 2003 would consist of two simultaneous versions. Eight students chose to enroll in the three-credit version that, in addition to meeting three days a week for lecture/discussion, required students to complete several homework activities (involving both programming and paper/pencil tasks), and a semester long research paper. An additional twelve students chose to enroll in the four-credit version that consisted of all of the requirements for the three-credit version of the course, plus one additional, regularly scheduled hour of "RoboLab."

Students in RoboLab selected their own three-member teams. While they were expected to attend the weekly RoboLab session, time was, for the most part, used as unstructured meeting time during which teams could demonstrate previous assignments and begin construction/coding of additional assignments. Each lab required the students to work as a team to code and construct the robot as necessary. Deliverables for each lab included group code, as well as individual write-ups by each member of the team. Students were told that it was expected they would need to spend a total of 5-8 hours per week in the RoboLab to be successful. Teams were provided with keys to the RoboLab facility and were told that they could use the room at any times that worked for their teams.

Activities

Participants in RoboLab completed seven different lab activities. The following are brief descriptors of the labs as well as a short discussion of student approaches and/or difficulties. Complete lab instructions for each lab are available online [6].

RoboLab #1 – Object Avoider (using the touch sensor)

Primarily a "starter" lab, the Object Avoider lab asked students to construct the basic double bumper robot from the Constructopedia™ (The large instruction guide the comes standard with the RIS). The students were to program the robot to serve as a simple stimulus/response robot. The robot was to progress through its environment until it detected that it had interacted with an object. At that point, the robot was to back off, turn to avoid the object, and continue on its way.

While students were free to interpret directions relatively freely, they were required to justify their control decisions. Most chose to use a combination of a random length "reversal" with a random length turn "away" from the encountered object (with a double bumper configuration it is possible to detect with slightly more precision "where" the obstacle is with respect to the robot).

RoboLab #2 – Line Follower (using the light sensor)

Designed to allow students to both write more complex leJOS code as well as introducing them to the light sensors, RoboLab#2 was a two-part lab. Students were told that their robot would be placed in a "monochrome" environment with a single color "in bounds" area and a contrasting "out of bounds" area. Their robot's task was to navigate around the environment while staying in bounds. When the robot detected that it was proceeding out of bounds it was to return in bounds in a "sensible" fashion. Robots were to account for both black on white and white on black environments. In the second part of the lab, students were asked to produce a robot that could navigate a monochrome environment by following a "smoothly curving line" roughly 1 inch wide.

The challenge was to understand how to handle "loss of line." That is, what do you do when the sensor no longer detected it was above a line? While most groups chose to pivot the robot around its center axis in increasingly larger arcs until the line was "reacquired" the most successful group chose to zig-zag back and forth along a line edge. By employing a constant "creep forward to the left until line is lost" then "creep forward to the right until line is reacquired, they managed to produce a robot that moved extremely quickly around tracks of several different shapes.

RoboLab #3 – Shape Tracer (using the rotation sensor)

Designed to allow students to built control structures that produced as much accuracy as possible in their robot, lab #3 introduced the use of the LEGO rotation sensor. This sensor's output is a voltage from one of four discrete values. As the input axle rotates, the output voltage changes. The voltage changes 16 times through a single 360-degree rotation. Students were to use this knowledge and a little bit of mathematics to create a robot which could be provided some integer N and some distance M in centimeters and would then traverse along the outer edge of an N-sided regular polygon with sides of length M.

Students were largely judged on the robot's ability to finish tracing the shape on or near its starting point. This turns out to be a non-trivial task. While one group was able to finish within 2 centimeters of its starting point regardless of shape and size, most groups were off by anywhere from 10-20% of M, and one group, when asked to produce a square with sides of 1 meter, managed to end the task well over 2 meters from the starting point.

RoboLab #4 – La Cucaracha (hill climbing, simulated annealing)

La Cucaracha was designed to blend skills/code created in the completion of labs 1 and 2. Students were asked to create a robotic cockroach by creating a robot that avoided obstacles and sudden bright lights – both stimulus/response actions – as well as actively seeking a dark space.

While the stimulus/response actions of the robot did not prove to be difficult, students took significantly different approaches towards the generation of behavior that caused the robot to "seek dark." Several employed a simplified "hill climbing" approach involving moving in any direction which their robot perceived as darker than their current location until they reached a location where any movement took the robot to a spot which was "brighter" than the current location. Others implemented an occasional "random walk" which attempted to avoid becoming stuck at a local maximum. One group even attempted to remember the location of the last local maximum and implemented a time interval during which the random walk had to produce a result that was "no worse." If the robot failed to do so during that time interval, it returned to the previous maximum and took a different random walk.

RoboLab #5 – SuckerBot (searching and simple planning)

This lab is based on the vacuum world problem repeatedly discussed in Russell and Norvig's AI text and was modified with permission from an activity originally written by David Musicant at Carleton College. Students were to construct a robot that could navigate a 3x3 grid searching for "dirt piles" (squares of construction paper). When the robot detected dirt it was to clean it up (emit a tone which signaled a human user to

remove the construction paper). When the robot had cleaned the entire world it was to return to a "home cell."

In early tests, students were allowed to start their robot in a known location (the home cell). In further testing the robots started in a known orientation (facing east) but did not know their specific location. They were, again, required to clean the entire grid and return home. While all students produced robots that succeeded at this later task, there were significant differences in how robots achieved this goal. One group determined its location first before performing any dirt sensing/cleaning activities. They did this by navigating to the home cell first (traveling northwest until they had worked their way into a corner) and then completing the original task. While this was a good implementation of "code reuse" it was not a particularly efficient solution. Other groups largely reversed this procedure by performing a semi-random traversal until they had determined all cells were clean, and then making a similar northwest run to home. Still other groups conducted fairly structured traversals that cleaned as they worked their way to the cell furthest from home (determining their location in the process) and then figuring out the most efficient way to clean the remaining cells on their way back to the home base.

RoboLab #6 – aMAZEing Bot (search, knowledge acquisition and planning)

In probably the most complex lab of the semester, students were asked to create a robot that could search a maze of unknown shape that was superimposed on a 5x5 grid. "Walls" in the maze were represented by black electrical tape, while "openings" were represented in red electrical tape. Through this technique, robots were able to keep track of the specific cell they were in by detecting the crossing of red lines. It also allowed for a maze that took up less space overall since robots did not have to worry about backing away from walls prior to turning. Robots began in the home square, navigated the world, determining the shape of the maze as they went. As soon as they located the "goal cell" they were to return to the "home cell."

Most students programmed a robot that used a slightly beefed up version of a depth first search. That is, upon entering any cell there are, in theory, three different directions in which the robot could leave the cell. Thus, treating each cell as a node in a search tree it has, at most, three child nodes. Known walls (outside walls) and discovered/previously-learned walls (maze walls) can reduce this down to zero, making the cell a terminal node. By employing a depth first search with an aspect of knowledge acquisition (there is no reason to attempt to expand a child node in a direction previously known to contain a wall) robots were able to efficiently navigate the maze as well as develop a known path to the home cell.

Final Projects

Each team proposed a final project that was to demonstrate one or more of the AI techniques they had learned during the semester. Descriptions of the final projects during the fall of 2003, including relative levels of completion, are as follows:

- *Tic-Tac-Toe Playing Robot* – Using the grid and programming techniques from the SuckerBot Lab, this group produced a robot that was "player 2" in a tic-toe-game. That is, it began by navigating the grid to determine in which cell the human player had placed its first piece (construction paper). Once it located this, it selected its responding move, and navigated to that cell in the grid, and signaled the human player to mark its move. It then awaited a signal from the human that it had placed its second piece, and then repeats the above process of locating and processing the human's move. While this group succeeded, the intelligence of their player was severely limited by the limited memory on the RCX bricks.
- *Mancala Playing Robots* – Mancala was an ongoing example throughout the classroom portion of the AI course. This team attempted to create two Mancala Playing robots placed on opposite sides of an oversized Mancala board. Each robot would determine from which pit it wanted to pick up stones, move to that pit, and signal the human "supervisor" to distribute the stones. Upon receiving a signal from the human that this action had been conducted, the robot would return to its "home base" and send an IR signal to the other robot including the previous move. This robot would determine its move and repeat the process. This project was simply too complex to allow for completion in the timeframe allowed. The team solved basic movement and communication issues, but was unable to produce robots that intelligently played the game, and an unresolved bug caused the agent to make each move twice before signaling the opponent that it was done.
- *Fax Machine* – Using a structural design from the LEGO materials, this team programmed two different "robots" to use IR communication to coordinate the movements of a "scanner bot" and a "plotting bot." While the resolution of the material to be scanned and reproduced was severely limited by the resolution of the light sensor, this group was able to create a relatively accurate "block plotter." Their real challenge was in the coordination of the two bots. They were unable to determine how to handle the lag between the scanner detecting that the pen should be raised/lowered and the plotter actually performing the action. When combined with a system that used bi-directional scans (the scanner made a pass from left-to-right, advanced slightly, and then scanned from right-to-left on the next pass) produced plots that had very jagged edges.
- *Obstacle Avoiding Search and Rescue Robot* – The final team produced another robot that started at a known location in a grid and searched the grid for a goal cell. However, the grid contained several obstacles in unknown locations and orientations. The challenge was to navigate the grid, avoiding obstacles, to find the goal cell and return home. This project was heavily influenced by the robotics competition being held in conjunction with this year's MICS conference.

Lessons Learned and Recommendations

For the most part, we were very pleased with the results of RoboLab. When asked if they would "do it again" and if they would recommend RoboLab to students taking the course in the future, 10 of the 12 students provided positive feedback. Having said that, there were clearly things that we would do differently in the future, and students completing the course were more than willing to point out ways in which they thought the course could be improved. This section will focus on several of the lessons that we learned in teaching RoboLab, and recommendations for those considering such a course. These issues include course structure, control issues, and equipment needs.

Course Structure

Student concerns about RoboLab fell relatively clearly in two camps. The first of these were issues surrounding the amount of time that they spent on RoboLab projects.

While it was our initial intention that students spend 5-7 hours a week on RoboLab tasks (including both scheduled course meetings and unscheduled team times) most students reported a figure about twice this. By students' own admission, part of the problem was that they enjoyed "tinkering" with robotic design too much. They would spend two hours trying to get a VaccuumBot that looked "cool" or that played the theme song to Legend of Zelda when it reached the home square on the grid.

However, they were also working fairly hard. The first four labs were completed one per week one right after the other, and the fifth lab actually was divided into two mini labs which were due in two consecutive weeks. Only labs six and the final project extended over multiple weeks. Thus, teams had very little flexible time, as there was always an upcoming deadline.

The other downside was that the instructor knew that he was working the RoboLab participants rather hard and, despite the fact they were earning an extra hour of credit for this work, he was reluctant to burden them to much more. As such, the quality and quantity of homework assignments made in the regular AI classroom suffered. On several occasions I convinced myself to delay or cancel a homework assignment because it meant that RoboLab participants would have to be working on two difficult coding assignments simultaneously.

So what is the solution? One solution is to drop back to a single offering style for the course. That would involve rolling RoboLab into the three-credit version of the course, or requiring that everyone enroll in the four-credit version. However, we do not feel that this is the solution that we will take. In order to put RoboLab into 3 credits we would have to eliminate some topics that we feel are important to cover. But on the flip side, we clearly have students who cannot afford the time required to take a four-credit version of this course.

Thus, for the fall of 2004 we will once again be offering two concurrent offerings of the course. However, in order to address the time concerns of RoboLab we will be backing off slightly on the quantity and length of the labs being required. This will be attempted through two independent actions. First, we need to re-evaluate the ultimate goal(s) of each of our previous labs and either eliminate some labs, or merge labs into a single lab ("killing two birds with one stone"). Second, we can, perhaps, reduce the programming time required of our teams by providing students with a calibrated, robot control package as discussed in the next section.

Robotics Control

The one thing that caused students the most difficulty during the course of the semester was achieving the fine-grained level of control over their robots that is needed to produce accurate results. The motors that come with the Mindstorms kits are highly sensitive to voltage, and a robot that travels perfectly straight during testing can travel with a significant curve one direction or another as the voltage in the batteries depletes. While the more successful groups learned to always work with a set of fully charged batteries to try to have reproducible results, even this was not enough to eliminate significant control problems in groups with otherwise good code.

Because of this several of the students from the Fall 2003 course have suggested, and are contributing code towards, a standardized Robotics Control package that would be built on top of the leJOS language. This package would provide the ability for programmers to run a calibration program prior to any runs of their robots. This program would allow for simple human user adjustments of voltage over individual motors until the robot was traveling straight given the current battery voltage. Robotics control code would then replace a series of calls such as:

```
motorA.setVoltage(9);
motorB.setVoltage(8);
motorA.forward();
motorB.forward();
```

with a single call to a helper method such as

```
RobotControlPackage.robotForward();
```

While the development of this package is expected to be non-trivial, it is expected to be doable with some part time, cooperative work of several students and the instructor, and be available for use by the time of the next offering of RoboLab (fall 2004).

Robotics Equipment

One of the criticisms of using the Mindstorms as a platform for robotics instruction is that you can't build anything serious with the basic RIS [1]. However, it was our observation that very little additional equipment is needed to produce *reasonable* kits that can complete labs focusing on most of the introductory AI topics. Other than the basic RIS,

the only additional equipment necessary for the labs provided in RoboLab are a rotation sensor and an additional light sensor. The double light sensor configuration is needed to perform runtime adjustments to the direction of the robot, and the rotation sensor can allow students to more accurately control angular turns of their robot. Since one or both of these frequently come bundled with the previously mentioned RISK instructors looking at adopting this approach may want to consider the purchase of the RISK rather than the RIS and the sensors separately. Frequently the costs are comparable, however, by purchasing the RISK you get "free," additional components (although the actual helpfulness of these additional components is debatable and depends on the particular bundle being offered). Having said all of this, every lab described in this paper was completed with a kit assembled for approximately $250 per kit – a cost far less than many of the more "serious" robotics kits.

The other concern that we continue to have about the Mindstorms platform is its limited onboard memory capacity. Until LEGO builds a Mindstorms kit that includes the ability to expand the unit's memory through devices such as Smartmedia or Flash Memory, students will be quite limited by the 32KB of RAM contained in the RCX. Attempts to perform any kind of serious search very quickly cause a memory dump due to overuse of the RAM. For example, the tic-tac-toe playing robot originally attempted to use the MiniMax algorithm to decide which move to make, but even with a game this small, the data structure(s) produced during execution of the algorithm would not fit on the RCX.

Conclusions

Despite the limitations of the Mindstorms' platform, and despite the initial difficulties, we were pleased with the initial results of RoboLab. While it was not without its faults, the lab provided students with the opportunity to have a lot of fun and participate in activities that motivated them. While a series of revisions will need to be made to find the right balance between fun, busy work, and actual learning through application of ideas from the classroom, the author firmly believes this balance exists and is looking forward to trying the whole process all over again during the fall of 2004.

References

1. Klassner, F., "A Case Study of LEGO Mindstorms' Suitability for Articifial Intelligence and Robotics Courses at the College Level," Proceedings of the 33rd SIGCSE Technical Symposium on Computer Science Education, pp 8-12.
2. Learning Styles and Strategies home page, http://www.ncsu.edu/felder-public/ILSdir/styles.htm
3. leJOS home page, http://http://lejos.sourceforge.net/
4. NQC home page, http://www.enteract.com/~dbaum/nqc/
5. RCXLisp users manual, http://www.csc.vill.edu/~klassner/csc4500/RCXLisp-Manual.pdf
6. RoboLab home page, http://www.cs.uni.edu/~schafer/robolab/

7. Thomas L., Ratcliffe, M., Woodbury, J., and Jarman, E., "Learning Styles and Performance in the Introductory Programming Sequence," Proceedings of the 33rd SIGCSE Technical Symposium on Computer Science Education, pp 33-37.

Acknowledgements

I would like to acknowledge Dr. Bart Bergquist and the entire Department of Computer Science at UNI for their initial and continued support of the construction and implementation of RoboLab.