A Web-based Programming Environment for LEGO
Mindstorms Robots

Aaron Garrett
Jacksonville State University
Jacksonville, AL 36265
agarrett@jsu.edu

David Thornton
Jacksonville State University
Jacksonville, AL 36265
thornton@jsu.edu

ABSTRACT
In this paper we discuss a web-based interface for programming a LEGO Mindstorms robot. The Mindstorms Internet Control Environment (MICE) allows students with web access to upload code to a Mindstorms robot and view its subsequent actions via a webcam. In addition, the environment provides users with the ability to collaborate with other users who are waiting to use the system.

Author Keywords
LEGO Mindstorms, telerobotics, mobile robot, web control.

INTRODUCTION
One of the most difficult aspects of computer science is learning to solve problems effectively. This one skill is crucial to the success of a computer science student. Traditionally, however, very little emphasis has been given to teaching effective problem solving. Once this topic is considered, several questions arise. For instance, what types of problems should be solved? While it is true that the general techniques of problem solving should apply across all domains, it is necessary to realize these techniques in a particular domain in order to gain proficiency [2].

Also, how do we extend individual problem solving techniques in order to apply them in a team-based environment? The answer to this second question bears heavily on computer science because so much of the industry relies on a team-oriented process. Teaching and giving students experience with solving problems as a part of a group is, therefore, of paramount importance.

In this paper, we present briefly an approach we have developed that deals with these issues. We describe an ongoing, evolving course in problem solving that makes use of the LEGO Mindstorms robot kits in order to give students a hands-on approach to real-world problem solving. We then emphasize our development of a system that allows students to collaborate as a team to program a LEGO robot across the Internet [3]. This system makes it possible for students to have regular access to the robot in order to facilitate their learning. This paper focuses primarily on the design and usability of the system’s client-side interface.

TEACHING PROBLEM SOLVING
As previously mentioned, teaching the skills associated with problem solving is a largely overlooked area of the computer science curriculum. However, it is of great importance for preparing students to work in the industry. Problem solving skills are always expected but rarely taught. Many courses claim to teach problem solving skills in addition to the principal topic. However, students and faculty often get lost in the details of the topic and fail to focus on problem solving. Students, then, are expected to “learn it on their own” with little or no preparation.

In order to alleviate this problem, we have developed a secondary component to an introductory programming course that focuses entirely on problem solving. In this component, we lecture about general problem solving techniques (such as finding a pattern or creating subproblems). We do not force students to learn a new domain wherein to solve problems. Instead, we give them recreational math problems (such as brain teasers and logic problems) that strengthen their understanding of the various problem solving techniques taught in class.

Additionally, since the course is computer science, we make use of the LEGO Mindstorms robot kits as an added application of the problem solving techniques. Students learn to write simple programs for the robot in order to solve basic problems posed to them. Programming the robot provides several benefits. First, students find the visual, aural, and kinetic feedback of a moving robot more engaging than that of a simple text-based program. Second, this increased interest motivates students to spend extra time on their assignments. Third, students can quickly test and debug their ideas. Fourth, the use of a real-world robot introduces students to interesting problems (such as unexpected variables) that they would not otherwise face.
Fifth, the students tend to think of the robot as an object, and this conception provides foundation for introducing the ideas of object-oriented programming [1].

We assign teams of 4-5 students to work together creating programs for the robot. This team aspect reinforces the need to clarify and communicate solutions. It also gives students the opportunity to learn from one another and to appreciate different approaches to solving the same problem. The teams are given part of the class period each week to collaborate on a particular problem.

The main drawback of this approach is that students have limited access to the robot. The cost of the Mindstorms kit ($200) prohibits us from allowing each student to take a robot home with them. To address this problem, we have developed a web interface that allows students to remotely program the robot and watch the results via a webcam.

This way, teams can refine their solutions between class meetings. We refer to this system as the Mindstorms Internet Control Environment (MICE).

THE MICE SYSTEM
The telerobotic web interface which provides students with the ability to control a LEGO robot uses a client-server architecture. This architecture allows transmission of NQC source code (the language we use for programming the robots), webcam images, queue scheduling information, user information, and user chat messages [3]. This system requires only that the user have a Java-enabled modern web browser with access to port 6620.

Server
The server for the system handles all requests for use of the robot. It maintains and manages a queue of users who are currently waiting to use the robot. In addition, the server must interface with a webcam in order to deliver constant visual feedback about the robot and environment. Also, it must handle any communication between two users (which provides online collaboration). Finally, and most importantly, the server must interface to the NQC compiler and robot in order to compile, download, and run programs.

Client
The client is the user’s interface to the system. It provides visual feedback about the state of the robot and environment. Additionally, it includes tools to help students use the NQC language as they write their programs. It also provides the ability for users to collaborate via chat messages while waiting on or controlling the robot. The client is implemented as a Java applet that is embedded in a web page. The remainder of this paper discusses the client interface in more detail.

THE MICE INTERFACE
When users first visit the MICE site, they must log in to use the system. Then, they choose the robot (if multiple robots are available) that they wish to control. (See Figure 1. Choosing the Robot) The interface specifies the expected time to wait for each robot, which may influence their decision.

The Queue Representation
After choosing a robot, the main interface applet is loaded. The top of the applet contains a panel that represents the queue of users waiting to control the robot. The queue is represented by a line of “stick men” icons. The user’s icon is always colored white. In addition, this panel provides information about the amount of time the user must wait for control of the robot. As the time to wait decreases, the panel’s background color changes from red to yellow to green. A countdown clock also sits at the top of this area. The countdown, likewise, is displayed in the title bar of the browser. This enables users to minimize the browser and still keep up with their place in line. The queue is automatically updated whenever new users enter or existing users leave the queue. Note that, at the moment, the queue size is limited to no more than 10 users.

Tools for Collaboration
By clicking on another user’s icon, a user may initiate a chat message session (much like Instant Messenger) with that user. This allows users to collaborate with one another on how to most effectively program the robot. This feature is an attempt to facilitate the team-based programming that the students are able to accomplish during the class period.

The remainder of the applet contains a tabbed pane with three primary tabs – Analyze, Implement, and Test. Most of the functionality for these tabs is available to all users in the queue. However, the ability to download/run/stop a compiled program and view an event log are restricted to only the user who is currently in control of the robot. Because of the likely possibility that a user may have to wait for long periods for control of the robot, we made...
every effort to provide them with as much functionality as possible to avoid unproductive delays. The following sections describe each of the three tabs in more detail.

**The Analyze Tab**
The Analyze tab (Figure 2. The Analyze Tab) allows the user to view the details of the current robot (such as sensors and motors) and environment (such as the current goal and obstacles). The user may click on the conceptual drawings of the robot and the environment to get additional details.

**The Implement Tab**
The Implement tab (Figure 3. The Implement Tab) allows the user to write and compile NQC programs that can later be downloaded to the robot. If compiler errors are generated, they are displayed below the code window. Clicking on these errors will highlight the line identified by the compiler. In addition, sample code is displayed on the left side in a drop-down list box which may be copied and pasted into their own code. Helpful links, which deal primarily with the NQC programming language, are displayed on the right side.

A useful feature that we added to the system is the ability of the user to include statements within their NQC code that essentially take “snapshots” of the robot at certain points during the execution. These snapshots are later made available in what we call an *event gallery*. The event gallery for a run is a compilation of all of the snapshots the user asked to be taken during that run, along with a timestamp for each snapshot. This makes it possible for the user to catalog and study the results of a run off-line. The event gallery can be downloaded by the user after a run is complete in the form of a PDF file or a compressed archive of images. These images correspond to each of the instances of the *snapshot* command executed in the source code.

When the user is in control of the robot and their code compiles correctly, they are given the option of downloading it to the robot. This automatically takes them to the Test tab.

**The Test Tab**
The Test tab (Figure 4. The Test Tab) allows the user to run and stop a current program on the robot. If the user has included code to generate an event gallery, the gallery may be downloaded from this screen after a run. The "Run" and "Stop" buttons, as well as the event galleries, are only available to the user who is currently in control of the robot. All other users may watch the robot's progress through a constantly updating webcam image. However, they have no control over the run.

**CONCLUSIONS AND FUTURE WORK**
The use of robotics in teaching computer science and, more specifically, problem solving is becoming more common due to the many benefits already described. However, the prohibitive cost of robot kits makes it difficult to give students extracurricular exposure to problems. We believe that the system described in this paper provides a simple, inexpensive solution to this problem. If students have access to the Internet, then they have the ability to study and explore the capabilities of the LEGO Mindstorms
robot. We feel that our system provides the basic functionality needed by an introductory programming student and is accessible to that group of users.

Future modifications to the system include a more adaptive client-side interface that detects the user’s connection speed and modifies the visual feedback accordingly to increase throughput. Also, we realize that many users may have Internet access through a university in which a firewall restricts access to the necessary port. We plan to modify the client-server communication to incorporate HTTP tunneling in order to alleviate the need for the additional port. Finally, we plan to provide a means to allow the user to “schedule” a time to use the system in order to eliminate the long and possibly unproductive waiting times.

REFERENCES