

# Adaptive Robot Control in Java: Programming LEGO Mindstorms® Robots

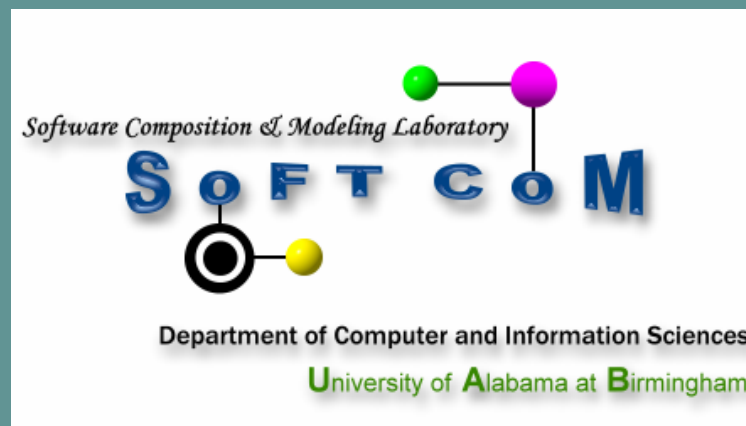
Summer Interns: Matt Ferguson and Erik Scott

Project Supervisor: Dr. Jeff Gray

Graduate Student Volunteers: Suman Roychoudhury, Alex Liu,  
Hui Wu, Jing Zhang, Danyu Liu, Jeremy Fisher, Vetria Byrd

*In cooperation with the Heritage Center*

<http://www.cis.uab.edu/heritage>



# Overview

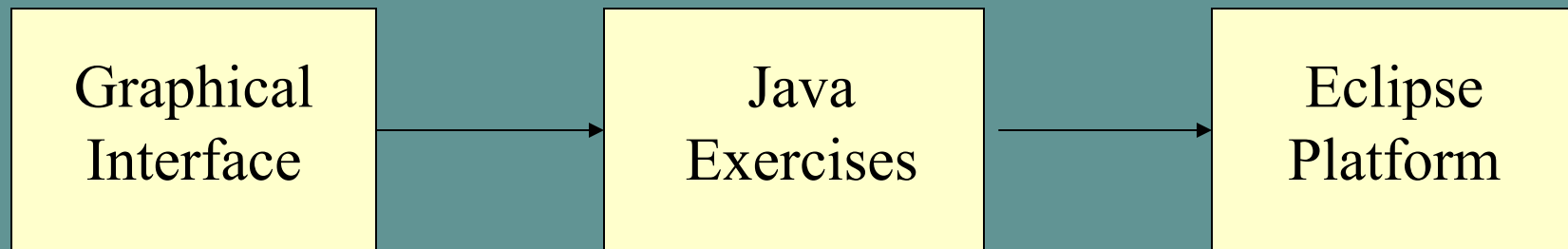
LEGO MindStorms® is a robotics kit that allows the user to program a variety of robots by utilizing the LEGO RCX. The RCX (shown below) is the computer “brain” of the robot, which can be reprogrammed by the user to perform many unique tasks.

By constructing and operating these simple robots, we gained a better understanding of computer programming languages and their function for controlling robotic control systems.

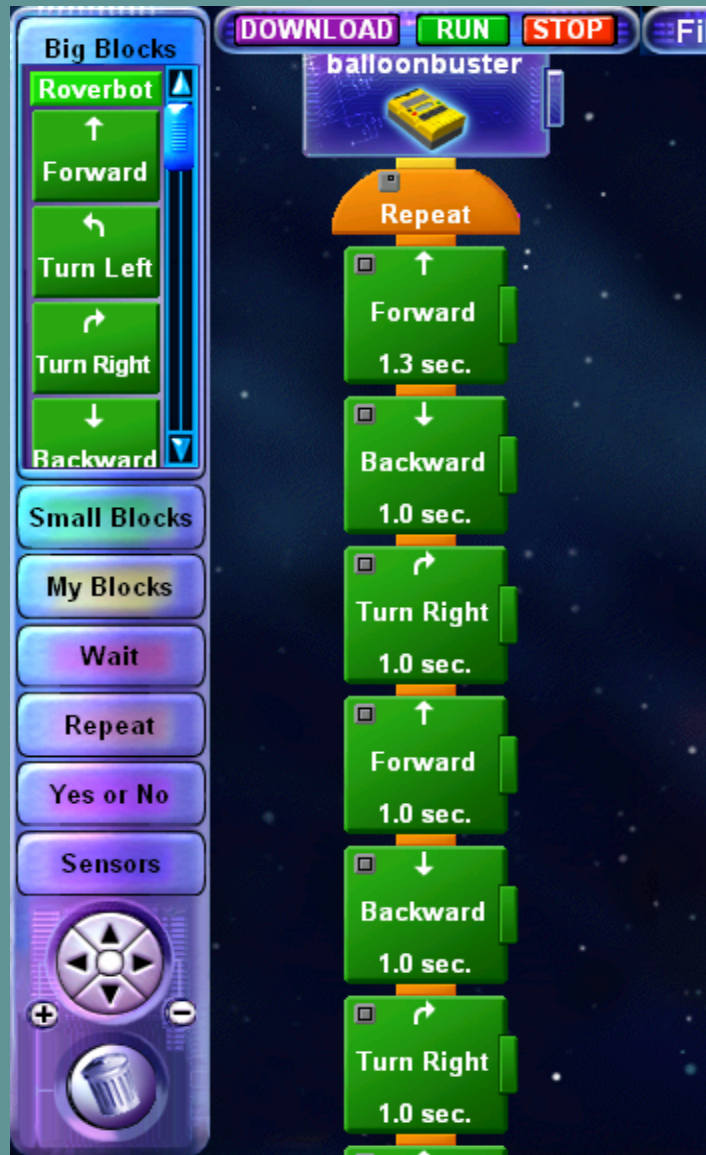


# Progression of the Projects

We started our 7-week internship by first learning to program the robots using the LEGO Robotics Invention Systems Graphical Interface, in order to become acquainted with the robots basic functions. After becoming familiar with our robots, we began to study the Java programming language through a series of small exercises. Then we were able to apply our knowledge to the programming of the LEGO RCX using Lejos within the Eclipse Platform.



# Graphical User Interface (GUI)



The LEGO Robotics Invention System GUI utilizes “blocks” of pre-programmed commands to run the robot. A programmer can simply arrange the blocks to create the desired actions.

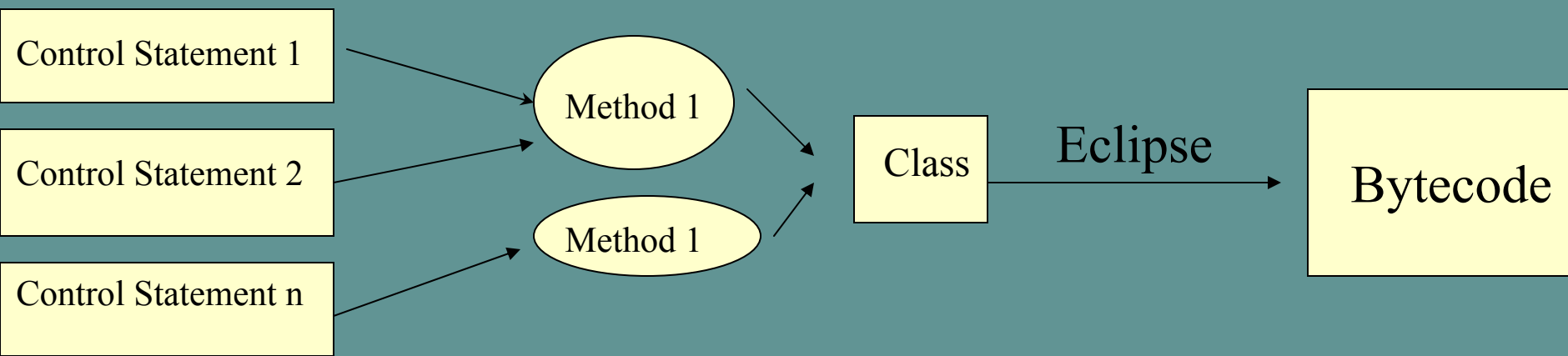
However, the GUI doesn't allow the user to modify the blocks, meaning the user has less control over the robot's functions. This approach is suitable for an introductory experience, but very limiting in ability to control the robot at a lower level.



# Java Programming Using The Lejos Library and Eclipse Platform

To make the LEGO robot programmer's task easier, Lejos, a library of Java control statements made specifically to control the RCX, was used. Lejos combines several basic Java control statements into a general method for movement, sensors, or other RCX functions.

The Eclipse Platform compiles Java-based Lejos code into robot-usable bytecode to transmit to the RCX. It also debugs the code as it is written, making error-correction much faster.



# Single Robot Projects

- ◆ Balloon Buster: A set of 10 balloons will be placed in a circle with the robot starting in the middle of the circle. The robot will know the diameter of the circle and location of the robot in the center. The winner is the robot that can burst the most balloons in 3 minutes. The robots can use a needle pin attached to the robot to aid in the bursting. This idea was borrowed from Southern Illinois University (Edwardsville).
- ◆ Line Tracer: This project is rather simple. A large/thick black line is drawn on white paper and the robot should follow the line. The line may actually return back to the starting position such that the traversal is infinite. The line can have a lot of interesting turns and curves.
- ◆ Kick the Can: This game is played on a white circular rink demarked by a thick black line at the circumference. Seven 12 oz. Cans of soda are placed in marked positions, arranged radially from the center. The goal is to remove soda cans from inside the rink as quickly as possible. This project was described by Cen Li (from MTSU) who adopted it from F. Martin's "Robot Explorations" book.
- ◆ Homing Pigeon: This is a very simple project in concept, but can be surprisingly difficult. The robot is asked to drive past a line that is 10 feet away from the start position. After passing the line, the robot must make a 180 degree turn, and then return as close as possible to the initial start marker. The winner is the robot which gets closest to the starting point. It is much harder than it sounds! This project was suggested by Lewis Patterson of Birmingham Southern.

# Multiple Robot Projects

- ◆ Sumo Wrestler: This project simulates a Sumo wrestling event. You must write code that keeps your robot within a circle, but tries to push the other robot out of the circle. The winner is the robot that remains in the circle after the other goes past the line. No holds barred (that is, anything goes in the code - all moves are legal but the robot hardware cannot be altered!) This idea was borrowed from Southern Illinois University (Edwardsville).
- ◆ Simon Says: This is a “Simon says” project that is similar to synchronized swimming. One robot is a master and sends commands to the infrared of the other robot. Both robots then perform the action in synchrony and return back to a state where the infrared sensors point to each other for another command to be issued. You must collaborate with the other student and work on this project together. This project was suggested by Tivadar Szemethy of Vanderbilt.

# Mini-Project : Kick the Can

```
Resource - KickTheCan2.java - Eclipse Platform
File Edit Source Refactor Navigate Search Project Soln: Serviceable Plugin Soln: JDT Speech Run leJOS Window H
KickTheCan2... X .classpath TermProject1.java Averag

public class KickTheCan2
{
    public static void main(String[] args) throws Exception {

        Motor.A.setPower(7);
        Motor.C.setPower(7);

        LCD.clear();
        TextLCD.print("He");
        Thread.sleep(2000);
        TextLCD.print("World");
        Thread.sleep(2000);

        TimingNavigator n= new TimingNavigator(Motor.C, Motor.A, 5.475f, 4.03f);
        n.travel(56);
        n.backward();
        Thread.sleep(2000);
        n.rotate(65);
        n.travel(54);
        n.backward();
        Thread.sleep(2000);
        n.rotate(65);
        n.travel(58);
        n.backward();
        Thread.sleep(2000);
        n.rotate(65);
        n.travel(66);
        n.backward();
        Thread.sleep(2000);
        n.rotate(65);
        n.travel(62);
        n.backward();
        Thread.sleep(2000);
        n.rotate(65);
    }
}
```

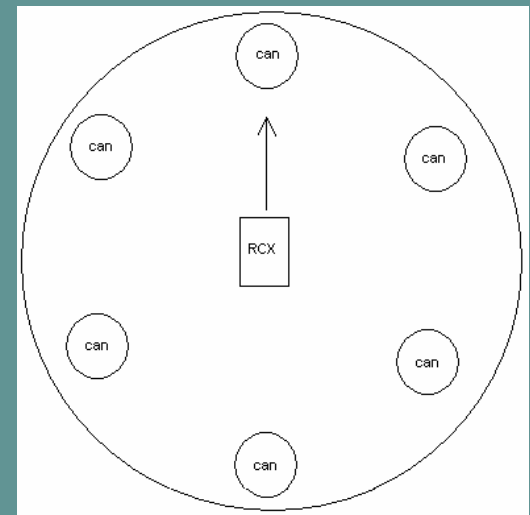
Screen shot of Erik's project

## • Problem

We had to create a program that could push 6 cans outside of an 80 cm diameter circle.

## • Solution

We had to calculate the distance and angle between the cans and outer circle, in order to determine how far to travel.



# Term Project : Maze Navigator

```
itter.java | Receiver.java | MazeOne.java X | CylinderVol.java | IntAvg.java
Sensor.S2.setTypeAndMode(3, 0x80);
Sensor.S2.activate();
// Store the value returned by the light sensor (S2) as int LightPercent
int lightPercent = Sensor.readSensorValue(1,1);

TimingNavigator n = new TimingNavigator (Motor.C, Motor.A, 8.13f, 3.95f);

// Used to initiate loop specific to current location
int i = 0;

// Used to determine when to break current loop before hitting maze wall
// and in final report on location of target.
float xCoord = 0;

// Used to determine when to break current loop before hitting maze wall
// and in final report on location of target.
float yCoord = 0;

// Initial scanning run. robot runs maze length wise (1,1 to 1,5)
while (lightPercent > 40 && i == 0)
{
    // Re-read S2 to determine if target located
    lightPercent = Sensor.readSensorValue(1,1);

    // If the robot detects the "target" (a large black dot), the robot
    // should gather current x,y position, display coordinates, and the
```

- Problem

The objective of this Term Project was to create a robot that could navigate a maze of set dimensions, without touching the walls, and find a black dot (position unknown) on the maze floor. From there, the robot had to return to the start of the maze, and report the dot's location.

- Solution

I took measurements of the maze, and made the robot travel a set distance to avoid walls. I used the

light sensor to detect the black dot, and the Lejos TimingNavigator method to determine the robot's current position. From there, the robot could calculate how far to travel, and in which direction, to return to home, and then displayed the coordinates of the dot on the LCD screen.

Screen shot of  
Matt's maze



# Problems We Encountered

- One problem we faced was that one motor seemed to overpower the other. This led to the robot veering off sharply to one side, even when both motors were given equal power.
- We faced interference problems with the IR ports on the robots, especially when trying to download to both robots from both IR towers at once. This often led to corrupted, unusable programs.
- The Timing Navigator, which determines the distance the robot travels in  $x$  seconds, was somewhat unpredictable. A drop in battery power made the robot run slower, thus altering the Timing Navigator values.

# Useful Experience

By participating in this program, we have learned much more about Computer Science and the art of computer programming. We've gained insight into the day to day life of a computer programmer, and into the field of robotics. We have also gained better study skills and self-reliance from this program due to it's focus on independent research.



# What Does the Future Hold?

After completing the summer program, Erik wishes to learn more about programming in general to decide if he really would like to pursue a career in it. Matt also enjoyed this opportunity, and is now considering further exploration in this field. Both are considering using their knowledge of Java and Lejos for entrance into science fairs, and perhaps other programs involving computer science.

# Special Thanks To:

- Mr. Jesse Higgs and the Heritage Center for their continued support throughout the project.
- Dr. Jeff Gray and the UAB Computer Science Department for his advice and guidance during the project.
- And to all those who offered us advice and support during this program.

