Workshop Schedule

• 8:30 - 9:00 am: Continental Breakfast
• 9:00 - 10:20 am: The Contest, Mindstorms Basics
• 10:20 - 10:40 am: Break
• 10:40 - Noon: Direct Control
• Noon - 1:00 pm: Lunch
• 1:00 - 1:30 pm: Brief Campus Tour
• 1:30 - 2:20 pm: Reactive Control
• 2:20 - 2:40 pm: Break
• 2:40 - 4:00 pm: Multi-tasking
Are Robots Descendant from Birds?

- They say that birds are descendant from dinosaurs.
- Are robots descendant from birds?
- Well, ours don’t lay eggs in their nests, but they just collect them and place them there.
The Contest

- In five minutes, a robotic bird tries to collect as many eggs as it can and return them to its nest.
- However, it has an opponent which is trying to do the same thing!
Other than nests, floor is white. Walls are 2” high and white.

12” square nest

1/4” high, 1” wide white lip around each nest

nest lip edges are smooth
An Actual Challenge World!
A Contest Game

- Two robots compete in trying to collect and return to their nests as many eggs (ping-pong balls) as they can.
- Each nest is either green or black.
- Each robot begins in their nest facing the other side, and has five minutes to search for eggs and collect them in their nest.
- On their side of the playing field will be eggs of their color (green or black) and on the opposite side eggs of the other color.
- At the end of five minutes the eggs in the nest will be tallied. Eggs of their color will be worth one point, eggs of their opponents color will be worth two points.
- You will be allowed to place three eggs of your color on your side anywhere you like (but outside the nest!), other eggs will be evenly placed by the judges on both sides.
- Highest point total wins! However ties are possible.
Game Rules

- The robot must fit within its nest area and can not extend beyond it. No limit on height.
- The robot is not allowed to intentionally disassemble.
- The robot is not allowed to intentionally damage or turn over its opponent. It may, however, intentionally get in the way of its opponent.
- Robots can only be build using the parts in the kit supplied by Alma College. No extra parts, glue, paper, etc!
- Eggs must be entirely within and touching the nest when the contest ends to score.
The Tournament

- Teams will be divided into divisions.
- Within a division, each team will play each of the other teams.
  - The number of divisions will depend on the number of teams.
  - Two to three divisions are expected.
- The two top teams within each division will advance to the playoffs.
  - Number of wins with the division is the first consideration.
  - The first tie-breaker is the winner of the head to head competition between the two teams.
  - The second tie-breaker is the total of all the team’s scores.
  - If there is still a tie, head to head rematches will occur until a clear winner emerges.
- The playoffs will be a single elimination tournament.
  - In case of a tie, the game will be replayed until a clear winner emerges.
Prizes

• Each contest and participant and coach will receive a competition t-shirt.
• The top three teams will receive plaques with the team member’s and coach’s names engraved on them.
Tournament Logistics

The tournament will be held on Friday, April 8, 2005.
The schedule for the day will be

- Refreshments 9:00 - 9:30 am
- Contest Overview Session 9:30-10:00 am
- The Contest! 10:00 - noon
- Brief Campus Tours  noon - 12:30 pm
- Lunch/Awards Presentations  12:30 - 1:30 pm

Some PCs will be available for last minute programming adjustments, but bring your laptop if possible.
Lego Mindstorms

- Lots of Lego pieces, plus
  - The RCX (the brains)
  - Motors (effectors)
  - Various types of sensors:
    - Touch (bump)
    - Light
    - Angle (rotation)
    - Temperature
    - Other third party sensors available

*Basic Roverbot Configuration from the LEGO Constructopedia*
The RCX brick has

- Three power outputs used to power motors and lights
- Three sensor inputs used to control active and passive sensors
  - Active: RCX supplies power
  - Passive: RCX just detects changes in resistance
- The RCX also has IR input/output
- Programs are downloaded by an IR tower to the RCX.
Firmware and Programs

- The Lego firmware provides an environment for the execution of programs on the RCX.
- Must be downloaded (just once) before programming the RCX.
- Other firmware solutions are available.
- Lego Firmware v. 2.0 permits
  - 5 separate programs
  - 10 threads
  - 32 global variables
  - 16 local variables per thread
  - Broadcast IR protocol
Programming Mindstorms

- Mindstorms ships with its own iconic programming language, but this is a very limited language.
- So other have developed for Mindstorms almost all the usual programming languages, including:
  - C
  - C++
  - Java
  - Ada
  - Lisp
- We will focus on NQC (Not Quite C).
- You can use any language for this contest.
NQC Programming Environments

- NQC was developed by Dave Baum and is now maintained by John Hansen
  - [http://bricxcc.sourceforge.net/nqc/](http://bricxcc.sourceforge.net/nqc/)
- For the Macintosh the IDE of choice is MacNQC
- For Windows the IDE of choice is Bricx Command Center (bricxCC)
Lesson 1: Direct Control (DC)

• In the Direct Control paradigm the bot doesn’t interact with the environment, but just moves around in it.
• So we need to know the basic NQC commands we can use to control bot movement.
Basic Motor Control

- Recall that there are three power outputs, and in NQC the motors attached to them are called:
  - OUT_A
  - OUT_B
  - OUT_C

- Basic motor control commands:
  - OnFwd( ... ) : run the ... motor(s) forward
  - OnRev( ... ) : run the ... motor(s) in reverse
  - Off( ... ) : stop the ... motor(s)

- Another useful command:
  - Wait( ... ) : pause the program execution for ... microseconds
task main () {
    OnFwd(OUT_A);
    OnFwd(OUT_C);
    Wait(400);
    Off(OUT_A);
    Off(OUT_C);
}

Drive forward for 4 seconds
task main () {
    OnFwd(OUT_A + OUT_C);
    Wait(400);
    Off(OUT_A + OUT_C);
}

*Drive forward for 4 seconds, v. 2*
A Problem

- Design a program which would direct the bot forward for 5 seconds, and then have it turn around and come back to its starting point.
- What’s the approach to this problem?
A Solution

```c
void main () {
    OnFwd(OUT_A);  // Move Forward
    OnFwd(OUT_C);
    Wait(500);

    OnRev(OUT_C);  // Turn Around
    Wait(200);

    OnFwd(OUT_C);  // Come Back
    Wait(500);

    Off(OUT_A);     // Stop
    Off(OUT_C);
    Off(OUT_C);
}
```
Setting Motor Power

• You can set an output power level by the SetPower command.
• Examples
  • SetPower(OUT_A, 4);
  • SetPower(OUT_A + OUT_C, 1);
• The first parameter is the output(s), the second a power level in the range 0..7.
• Affects motor torque more than actual speed.
Learn by Doing

- Write, download, and execute an NQC program that will drive the bot around in a square.
- How about one of these shapes instead?
A Fancy Solution: Inline Functions

```cpp
task main () {
  repeat(4) {
    moveForward();
    turnLeft();
  }
}

void moveForward() {
  OnFwd(OUT_A + OUT_C);
  Wait(500);
  Off(OUT_A + OUT_C);
}

void turnLeft() {
  OnFwd(OUT_C);
  OnRev(OUT_A);
  Wait(100);
  Off(OUT_A + OUT_C);
}
```
task main () {
    while (true) {
        moveForward(500);
        turnLeft(100);
    }
}

void moveForward(int time) {
    OnFwd(OUT_A + OUT_C);
    Wait(time);
    Off(OUT_A + OUT_C);
}

void turnLeft(int time) {
    OnFwd(OUT_C);
    OnRev(OUT_A);
    Wait(time);
    Off(OUT_A + OUT_C);
}
In the Reactive Control paradigm the bot controls its actions in response to input from its sensors.

Two types of sensors come with the Mindstorms kit:
- Bump Sensors
- Light Sensors

Recall that there are three sensor connections, and in NQC the sensors attached to them are called:
- SENSOR_1
- SENSOR_2
- SENSOR_3

More than one sensor can be attached to a connection.
Using Sensors

- To use a sensor we must first define its type using `SetSensor`.
- Examples:
  - `SetSensor(SENSOR_1, SENSOR_TOUCH);`
  - `SetSensor(SENSOR_2, SENSOR_LIGHT);`
- By default, touch sensors return 0 or 1 (1 if currently pressed).
- By default, light sensors return a value in the range 0..100, with 0 meaning no measured light.
- Other return modes are possible, see the NQC manual for details.
task main () {
    SetSensor(SENSOR_2, SENSOR_TOUCH);
    
    OnFwd(OUT_A + OUT_C); // Move Forward
    
    while (SENSOR_2 != 1) // Until a Bump
        
    Off(OUT_A + OUT_C); // Then Stop
}
RC Example: Light Sensor

task main () {
    int startLevel, currentLevel;
    int tolerance = 5;

    SetSensor(SENSOR_2, SENSOR_LIGHT);
    startLevel = SENSOR_2;

    OnFwd(OUT_A + OUT_C); // Move Forward

    currentLevel = SENSOR_2; // Until Darker
    while (currentLevel > startLevel - tolerance)
        currentLevel = SENSOR_2;

    Off(OUT_A + OUT_C); // Then Stop
}
Learn by Doing

- Design a program that would have the bot wander around its world, backing up and turning a bit whenever it runs into anything. It should continue to explore its world until you stop it.
- Can you design the program so that it turns one way if the left bumper is pressed, and the other if the right bumper is pressed?
Lesson Three: Multi-tasking

• We want the bot to patrol back and forth until a bump sensor is pressed.
• Why doesn’t the solution on the next slide work?
A Proposed Solution

task main () {
    SetSensor(SENSOR_2, SENSOR_TOUCH);
    OnFwd(OUT_A);
    while(SENSOR_2 != 1) {
        OnFwd(OUT_C);
        Wait(400);
        OnRev(OUT_C);
        Wait(200);
    }
    Off(OUT_A + OUT_C);
}

This doesn’t work!
A Real Solution to our Problem

```c
#include <task.h>

task main () {
    SetSensor(SENSOR_2, SENSOR_TOUCH);
    start patrol;
    while(SENSOR_2 != 1) {
        stop patrol;
        Off(OUT_A + OUT_C);
    }
}

task patrol () {
    OnFwd(OUT_A);
    while(true) {
        OnFwd(OUT_C);
        Wait(400);
        OnRev(OUT_C);
        Wait(200);
    }
}
```
NQC Tasks

- The RCX supports up to 10 concurrent threads of execution.
- Use the `start` and `stop` commands to control task execution.
- Be careful how tasks use the RCX’s resources
  - Shareable resources: the sensors
    - One task’s reading of them does not affect what other tasks are doing
  - Non-shareable resources: the motors
    - If more than one task is trying to control the motors, unpredictable results will occur
    - This is an example of what in CS we call *race conditions*
Learn By Doing

• Modify the patrol solution so that whenever the bot’s bump sensors are pressed instead of stopping it spins around and then goes back to its patrol.

• More advanced problems:
  • Each time the bot is bumped it spins around one more time.
  • As above, but with a maximum number of times the bot can be bumped before it just stops.
  • As above, but that max number is entered into the bot by pressing its left bumper that many times, followed by a right bumper press. Then the bot begins its patrol.
Thank you

- If you have questions, please feel free to contact us
  - John Hawley (hawley@alma.edu)
  - Myles McNally (mcnally@alma.edu)