The Turtle Laboratory Sequence

LMICSE Workshop
June 14 - 17, 2005
Alma College
The Turtle Laboratory Sequence

- Designed to teaching programming fundamentals
- Based around the “Turtle,” a LOGO-like abstraction from the details of motors and sensors
The Turtle Laboratory Sequence

- Original work was done by Scott Anderson (Wellesley) with support from Frank Klassner (Villanova)
  - Coded the Turtle software
  - Created drafts of seven CS 1 oriented labs

- Current work by Myles McNally (Alma)
  - Simplified somewhat the Turtle software
  - Twelve labs oriented towards CS 1
  - Each lab was used in a fall, 2004 CS 1 course
  - Each lab has now been refined based on that experience and videos added
The Turtle’s Basic Abilities

- Motion
  - Move forward or backward (timed or continuous)
  - Turn left or right (timed or based on degrees)
    - Can calibrate turning time required to turn n degrees
  - Stop
- Events
  - Can respond to events
  - Simple event queue model
    - NONE (queue is empty), RIGHT, LEFT, BOTH, VIEW, PGM, RUN
- Singing
  - Can play a tone for a duration
The Physical Turtle (1)

- Many designs could be used
- Requirements are
  - Differential drive
    - Left motor in motor port A
    - Right motor in motor port C
  - Left and right front mounted bump sensors
    - Left touch sensor in sensor port 1
    - Right touch sensor in sensor port 3
The Physical Turtle (2)

- The design we recommend is based on designs from the Constructopedia
  - The Roverbot Driving Base (p. 12)
  - The Wheel Sets (p. 17)
  - The Double Bumper (p. 30)
Then make the following modifications
- Remove the front (smaller) set of wheels
- Add a slider to the bottom of the bot between where the front wheels had been
- Move the RCX itself further back on the base, so that its weight is more centered over the remaining wheels
- These changes improve the exactness of turns
The Current Turtle Laboratories

- Java-based (in particular LejOS)
- Covers topics found in a modern, object oriented CS 1 course
  - Basic types and expressions
  - Flow of control
  - Classes and methods
  - Arrays, stacks and lists
  - Interfaces
  - Inheritance and abstract classes
  - Polymorphism
- Loosely follows the topic order in the Lewis and Loftus text, but could be used with almost any object oriented text.
Turtle Laboratory Topics (1)

- **Sequential Control**: Use straight-line code and a "Turtle" robot to move in a few geometric figures.
- **Variables and Expressions**: Use more advanced code and variables to create more interesting shapes.
- **Methods**: Use methods to separate code into parts and also use the *Random* class.
- **Methods with Parameters / Scope**: Use methods with parameters and returns, instance variables, and the *Math* class.
- **Classes**: Define a class that allows musical notes and rests to be represented and played back by the RCX.
Lab 4, Task 5

QuickTime™ and a MPEG-4 Video decompressor are needed to see this picture.

Right Angle Random Patrol
Turtle Laboratory Topics (2)

- **Event Driven Programming**: Work with the basics of event driven programming and focusing on using decision structures to respond to events.

- **Loop Control Structures**: Work with each of the loop control structures in Java in the context of event processing.

- **Using Interfaces**: Define interfaces then implement them to run races and to draw figures with the robots.

- **Array Structures**: Use arrays to record inputs from the user and then traverse a course using the recorded values to know when to move and when to turn.
Lab 6, Task 3

Use Events to Drive the Turtle

QuickTime™ and a MPEG-4 Video decompressor are needed to see this picture.
Turtle Laboratory Topics (3)

• **Navigation**: Implement a navigation interface which allow the Turtle to go to positions in its world, then input a series of positions and have the Turtle visit them.

• **Inheritance**: Define several classes that handle notes and rests, and an abstract class that each is an extension of.

• **Sorting and Polymorphism**: Use polymorphism with various sorting algorithms.
Lab 12, Task 2

QuickTime™ and a MPEG-4 Video decompressor are needed to see this picture.

Travel to Points Sorted by Horizontal Position
public class Task1 {
    public static void main(String args[]) {

        Turtle.forward(2000);  // travel out and back
        Turtle.turn(1000);
        Turtle.forward(2000);

        Turtle.stop(); // stop all motors
    }
}

Travel forward, turn around, and travel back (see Lab 1)
public class Task2 {
    public static void main(String args[]) {

        Turtle.calibrateTurn(360, 3000); // calibrate

        Turtle.forward(2000);  // travel out and back
        Turtle.left(180);
        Turtle.forward(2000);

        Turtle.stop();         // stop all motors
    }
}

*Travel forward, turn around, and travel back using calibration (see Lab 2)*
public class Task3 {
    public static void main(String args[]) {

        Turtle.forward();  // move forward
        int event = Turtle.nextEvent();
        while (event != Turtle.RUN) {  // until RUN button
            event = Turtle.nextEvent();  // is pressed
        }
        Turtle.stop();  // stop all motors
    }
}
Hands-on Time!

- **Exercise 1**: Craft a program that will have the Turtle move in a square shape using a loop which contains one `forward` command and one `turn` command.
- **Exercise 2**: Improve your solution to include calibration, replacing the `turn` command with a `left` command.
- **Exercise 3**: Craft a program that uses the Turtle’s `event queue` to have the Turtle move forward, turning away from obstacles when necessary. That is, when the left bump sensor is pressed, back up and turn right, and likewise for the left bump sensor.
- **Exercise 4**: Craft a program in which you **enter a number** by pressing the right bumper that many times, and then have the Turtle travel around in a square shape that many times. Signal the end of your input by pressing the left bumper. The Turtle should then begin to move.