CS148
Building Intelligent Robots

Week 2 : Introduction to NQC *Out: 5 Feb 2001*

Preliminary Tasks *before 7 Feb 2001, 1pm*

The purpose of this week's lab and project is to provide an introduction to Not Quite C (NQC), the primary language you will use this semester for programming your robots. Before coming to lab, make sure you have:

- read chapter 3 of Baum
- built a robust tankbot with two treads, one motor controlling each tread (see p. 83 of Baum for inspiration)

Lab 2: Introduction to NQC *7 Feb 2001*

After completing the reading, work through the following problems with your partner to gain some experience with the concepts presented there and to become familiar with NQC.

Step 1

You will need to know how to make the robot rotate to a specific angle. It may take several attempts before you get the desired result.

Some general reminders about using NQC in the MSLab:

- When you are using straight NQC in the MSLab, you can use the JETEYE IR ports. However, if you want to use the NQC Command Center, you must attach the IR tower provided with your kit.
- To access your home directory, go to the Start Menu, select Run, and then type `\\godzilla\<login>`.
- NQC is located in `\\godzilla\course\cs148\nqc`. To run it, drag `nqc.exe` into a command window. Then type in any options (for example, the syntax for downloading a program is `nqc -d <filename>`) and drag over the file to download.

Task: Make the robot turn 90 degrees from it's starting position.

Step 2

In this problem, you will learn how to use two important features of NQC, the timers and the datalog. Refer to Baum for details, but here are a couple of important reminders:

- **Timers:** You start the timer by calling `ClearTimer(x)`, where `x` is the number of the timer you're using (1-4). This automatically restarts the timer.
- **Datalog:** The datalog is one of the only debugging tools in the standard NQC package. Other GUIs (like the RCX Command Center) offer real-time variable, motor, and sensor feedback, but the datalog is the only way to store and analyze data while the robot is running (using just NQC).

Task: Make the robot drive in a square where the length of each side is one foot. Have the robot pause at each corner after the 90 degree turn and add the timer value to the datalog. Additionally, datalog the value at the starting point - you should have five values total. Finally, have your robot play a sound after the last turn.

Step 3

For the final task, you will learn how to use the touch sensor. Here are some pointers:

- You will have to set the sensor type before you use it. If you want to attach the touch sensor to input 1 you need to type `SetSensor(SENSOR_1, SENSOR_TOUCH)`.
- The sensor is a simple binary switch — the statement

  ```
  if (SENSOR_1 == 1) {
    OnFor(OUT_A + OUT_C, 200);
  }
  ```

  will make the robot go forward when the touch sensor is pressed.

Task: Have your robot move in a circle when the touch sensor is pressed.

Project 2: Introduction to NQC *due 14 Feb 2001, noon*

Specification:

The exercises in this lab have covered most of the basics of NQC programming - syntax for tasks, subroutines, and motor control. Some of the loop controls (for, until, while) you will have to learn how to use on your own - for the most part, they are identical to your favorite high-level language. Refer to Baum or any of the numerous online references for the exact syntax. For this week's project, you will experiment with both of your touch sensors, and implement a simple obstacle-avoiding robot. We recommend using a tankbot - you will already have built one for lab - for ease of turning (you will need to navigate around several objects). Your robot will be placed in the center of an "arena," surrounded by several layers of obstacles such as walls and boxes of different shapes. You will not know the placement of the obstacles beforehand, so it will be impossible to simply program the robot to avoid without using the sensors. In addition, your robot should be able to find it's way out of the enclosure of obstacles more than once (we should be able to pick it up after it clears the obstacles, put it back in the center facing a different direction, and it should be able to find its way out again). Finally, your robot should assume that it has cleared all the obstacles if neither bump sensor has been activated for 7 seconds, and the program should end (motors off).

- How are you going to use your sensors? Do you want two front sensors that can differentiate between left and right, or a front and rear sensor?
- You should make good use of abstractions when coding the turnRight() or turnLeft() functions (called when the bumpers are toggled).
- Consider how the robot is going to account for corners - this is tricky, and very important.

Paper Handin:

Be sure to discuss the following items in your writeup:

- Briefly explain the design of your car (tankbot/wheelbot/yourcrazynewbot) and any special features you have implemented (differential wheelbase, etc.).
- Explain in detail your choices for bumpers, and the underlying code.
- Give a general overview of the flow of control of your program. Don't forget to mention any subroutines or tasks that run simultaneously.
- Be sure to include explanations of any nontrivial code, or special procedures.

Grading: You will be graded on equally on the functionality of your robot (50%) and on your report (50%). The robot will be graded according to how effectively it circumnavigates the obstacles. For example, it is a better design choice to make a robot that hits the obstacles more times but has a better idea of the size/shape of the obstacle than it is to make a robot that, after hitting the obstacle once, attempts to move around it without any idea of it's size/shape. In addition, your robot should be able to find it's way out of the enclosure of obstacles more than once.