# Introduction to NXT Programming Course

## Index

<table>
<thead>
<tr>
<th>Topic</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Running a Robotics Classroom</td>
<td>5</td>
</tr>
<tr>
<td>Teacher as a Facilitator</td>
<td>3</td>
</tr>
<tr>
<td>Teacher’s Notes - What is a Robot?</td>
<td>2</td>
</tr>
<tr>
<td>Teacher’s Notes – Introduction to the NXT</td>
<td>2</td>
</tr>
<tr>
<td>Teacher’s Notes – Introduction to Programming Robots</td>
<td>2</td>
</tr>
<tr>
<td>Teacher’s Notes – Moving Straight</td>
<td>2</td>
</tr>
<tr>
<td>Close Shave Challenge</td>
<td>1</td>
</tr>
<tr>
<td>Teacher’s Notes – Turning</td>
<td>2</td>
</tr>
<tr>
<td>The Maze Challenge</td>
<td>1</td>
</tr>
<tr>
<td>Teacher’s Notes – The Touch Sensor</td>
<td>2</td>
</tr>
<tr>
<td>The Vacuum Challenge</td>
<td>1</td>
</tr>
<tr>
<td>Teacher’s Notes - The Ultrasonic Sensor</td>
<td>2</td>
</tr>
<tr>
<td>The Walled Maze Challenge</td>
<td>1</td>
</tr>
<tr>
<td>Teacher’s Notes – The Light Sensor</td>
<td>2</td>
</tr>
<tr>
<td>The Intersections Challenge</td>
<td>1</td>
</tr>
<tr>
<td>Teacher’s Notes – Introduction to Loops</td>
<td>2</td>
</tr>
<tr>
<td>The Sentry Challenge</td>
<td>1</td>
</tr>
<tr>
<td>Teacher’s Notes – Obstacle Detection</td>
<td>2</td>
</tr>
<tr>
<td>The Sentry Challenge 2</td>
<td>1</td>
</tr>
<tr>
<td>Teacher’s Notes – Line Following</td>
<td>2</td>
</tr>
<tr>
<td>The Slalom Challenge</td>
<td>1</td>
</tr>
<tr>
<td>Teacher’s Notes – The Grand Challenge</td>
<td>2</td>
</tr>
<tr>
<td>The Obstacle Course</td>
<td>1</td>
</tr>
</tbody>
</table>
Running a Robotics Classroom Checklist

- Logistics/Installation
  - Acquired/reserved computers – one per group of 2-4 students
  - Installed LEGO® MINDSTORMS® Education NXT Software on each computer
  - Tested that each student computer can access the iCarnegie Learning Management System and open class documents
  - Purchased NXT rechargeable batteries and chargers, or purchased 6x AA batteries for each NXT (plus spares!)

- Pre-Lesson
  - Assigned students to teams of 2-4
  - Built Robots
  - Established class rules regarding the modification of robots and other resources that are shared among multiple groups/class periods
  - Have robots built by student volunteers (recommended)

- Teacher Preparation
  - Scope and sequence of Robotics activities chosen
  - Read through Teacher Notes for each lesson in the sequence
  - Step through the student steps for each activity if possible, or even do the activity yourself if you have time

- Lesson
  - Introduce students to the lesson using the Lesson Modules
  - Direct students to the appropriate starting point at the beginning of the lesson, and let them begin.

- End of Class (each period)
  - Return the robot to the original state (remove attachments; undo any changes you made to the gears, wheels, etc.)
  - Clean up loose LEGO parts (use the sorting trays and maps)
  - SAVE ALL PROGRAMS
  - Assign homework if appropriate

- End of Lesson
  - Collect worksheets to correct
  - Hold final discussion to answer any student questions
  - Administer Quiz
  - Move on to next lesson

- End of Project
  - Select one or more End of Project Activities
  - Schedule milestone dates – outlines, drafts, final presentation/competition, etc. depending on the specific activity chosen
  - Assign the selected activity at least 2-3 weeks before it is due
  - Prepare classroom or venue for final presentation or competition if applicable
  - Hold final presentation/competition
Running a Robotics Classroom Notes

- Logistics – Computers
  o Each group will need access to a minimum of one computer for best results. This computer must meet the minimum system requirements for both the NXT Programming Software (see Programming Software packaging for details) and the LMS software system.
  o If additional machines are available, you may consider two computers per team. This will allow the group to run the NXT Programming Software on one computer, and view the lesson instructions on the other; they can be used together on the same computer, but this separation makes it easier to see both at the same time.

- Batteries and Power Management
  o The 9797 Base Set includes one rechargeable Lithium-Ion (“Li-Ion”) battery per set; however, its use is optional, and a charger may not be included by default.
    ▪ You may elect not to use the rechargeable battery, and use six AA batteries per NXT instead. These will need to be replaced (or recharged, if you are using rechargeable AA cells) periodically as they are used up.
    ▪ You may purchase chargers for the Li-Ion NXT battery from your LEGO Education distributor, or you may already have a compatible AC adapter/charger from a previous LEGO product. Check the voltage specifications for the adapter before attempting to use it with the NXT rechargeable battery!
  o Use of the rechargeable battery is recommended for several reasons:
    ▪ The NXT Li-Ion battery can typically be used (under normal conditions) for an entire school day without recharging. Lower-capacity AA batteries may not last as long.
    ▪ The Li-Ion battery will generally provide the same amount of power to the NXT motors the whole time until they run out of power. Alkaline AA batteries will provide more power when they are fresh, and provide less and less power as they run down – this will contribute to inconsistent robot movement over the course of the battery’s life. For many robot navigation activities, consistent movement is crucial, and the variability of alkaline battery power can prove a frustrating obstacle.
  o If you get one charger for each battery, you can simply plug them all in to charge overnight
    ▪ The batteries will monitor their own power levels and prevent overcharging.
    ▪ This may require the use of power strips to get enough plugs for all the chargers, depending on the layout of your room and the number of robots you have.
    ▪ Check with your facility’s management to make sure that your room is able to handle the electrical current from a number of batteries charging simultaneously.
• Student Teams
  o All the Robotics activities are designed to be done by students working in teams. Teamwork is a crucial skill in the modern workplace, and the challenges of the Robotics activities lend themselves to group solutions.
  o Quizzes and exercise portions of worksheets are exceptions to this rule. They are designed to be done by individual students for assessment purposes.
  o Students should be formed into teams of 2 or 4 in most cases. Odd numbers of students on a team can often lead to problems with one student being left out and not doing anything. Groups larger than 4 are generally too large for all the students to have something important to do.
  o Unisex teams may be preferable, depending on the age range of the students.
• Sharing Sets and Modifying Robots
  o All of the Robotics lessons are based around the use of the LEGO MINDSTORM robots. The beauty of the MINDSTORM robot kit is that you can build multiple robots from the exact same kit.
    ▪ Each model takes between 20 minutes and 2 hours to build, depending on the experience level of the builder.
  o Which robot for which course –
    ▪ The Introduction to NXT Robot Programming and the Advanced NXT Robot Programming course will use the REM Bot.
    ▪ The Introduction to Robotics Engineering will use the Taskbot Robot. You will find building plans in the iCarnegie LMS.
    ▪ The Robotics Engineering 2 will use a variety of robots that will be designed by the student teams.
    ▪ The Introduction to ROBOTC course will use the REM; students may modify the robot during the course.
    ▪ The Advance ROBOTC course will begin with the REM robot; students will modify the robots as they move through the course.
  o Robot Logistics
    ▪ Each student team will need one robot.
    ▪ In most cases, the model should be pre-built by student volunteers before the start date of the in-class activities. It is not recommended that the students spend class time building the robot from scratch, unless this activity aligns with your curriculum goals.
    ▪ Once built, the robot base will generally not undergo any major modifications (however, there are a few activities where minor changes are made).
  o As sensors are introduced, various sensor attachments will be built and added to the base model.
    ▪ Instructions are provided for these attachments in the lessons that use them, and in the Building Instructions section of each lesson.
    ▪ Attachment building is short, and can be done in class if desired.
    ▪ The attachments only need to be built once. If you have multiple class periods that share the robots, this means that only one will need to build them.
    ▪ Some educational theory suggests that the act of building the attachments may lead to a better understanding of how the robot works; you may want to spread around the opportunities for different class periods to build the different sensor attachments.
  o Some activities will require minor modifications to the robot base.
• Students must be reminded to return the robot to the original configuration before leaving class, otherwise subsequent periods will start with robots that are not correctly configured for the beginning of the activity.
  ○ Advanced activities will require significant modifications to the robot. This will require a different allocation of the robot resources.
• Multi-day challenges work best when the students are able to change various physical aspects of the robots.
  • This will cause conflicts between different class periods which must share the same robots, if one group makes permanent changes to the robot that affect the way the shared robot base runs for the other group.
  • Being able to engineer a solution to a problem through a combination of physical and programming methods is a vital part of the learning experience, and should be supported if at all possible.
  • Having additional parts available for student use can help with this problem. Groups can use “extra” parts to build attachments, and not modify the main robot base. Additional parts can be purchased through your distributor, or often collected through donations from parents in the school who have unused LEGO elements at home.
  • Alternately, groups in different periods that share resources can be encouraged to “negotiate” a solution that is acceptable to all parties. Communication with peers and colleagues is another important skill that is vital in the workplace.
• Which robot?
  ○ Robot Educator Model (REM) – This is the base model for the 9797 Base Set itself, and is also featured in the sample programs found directly in the Programming Software. Its design is simple and easy to build.
• Structure and Pacing of a Lesson
  ○ Lessons are designed to be multi-day activities that students follow at their own pace (within limits).
  ○ Every lesson comes with a multimedia slideshow that helps guide the student through the unit.
  ○ Class discussion is appropriate and encouraged whenever students have questions or issues that are relevant to the activity or subject. The role of the teacher in such discussion is not to judge, but to facilitate productive discussion. When needed, guide students toward the right answer by asking questions that will open avenues of discussion toward a successful outcome.
  ○ Students learn at different paces within groups, it is important to make sure that all students are active, up to date, and contributing within their groups. It is just as detrimental to a student’s learning to do all the work (and not distribute the work efficiently) as it is for a student to be idle.
  ○ Some groups will finish early. There are additional activities available for these groups in the “Continue” and “Extension” sections of lessons.
• Assistants can help
  ○ Encourage students to both give and seek assistance from other groups in appropriate ways. There is much to be gained for all involved when one group of students helps another group to understand a concept that was holding them back.
- The group that did the explaining reinforces their own knowledge, and gains a sense of pride in their accomplishment.
- The group that received assistance is no longer stuck.
- The instructor did not have to spend the full amount of time working with the stuck group, and instead had that time available to help another group who might have been having problems.
  - Helping is not the same as sharing answers.
    - Someone providing assistance should not simply give the solution to the person or group they are helping. Simply giving answers does not help the receiving group to understand the concept any better, and constitutes a form of academic dishonesty.
    - Discussing a concept, clarifying directions, checking calculations, comparing programs, and critiquing an approach or argument are all great ways to build understanding and solve problems that do not involve giving answers.
  - Sometimes a group will come up with a particularly innovative or effective way to solve a problem. Allowing the group to share their findings with the class will both allow them to take pride in their accomplishment and help the class to work better. This is similar to the function of professors giving seminars at the university level.

- What do I need to know before I can start teaching?
  - Review the material for the lesson.
    - Know the general order of steps and where students should be at the end of each class period.
    - Know the important points that you want students to notice.
    - Practice the lesson by completing it before you teach the lesson.
  - Know how to operate the NXT brick (including the basics of the NXT Software)
Teacher as Facilitator

Traditionally, teachers have disseminated knowledge through direct means such as lectures, notes, summarizing text material, etc. While convenient, “teaching by telling” builds only knowledge. It does not address the need for a framework of understanding that binds facts together and lets students come away with true insight into the material which they are studying. Students must gain this understanding in order to master the material, but no amount of lecturing, quizzing, or drilling can force it into being.

By contrast, teachers acting as facilitators for discovery strongly contrast this traditional teaching style and puts students in the center of the educational process.

Key ideas for teachers as facilitators:

- Facilitating learning involves striking a delicate balance between allowing students enough freedom to explore the issues, and providing adequate direction and support to ensure that they can succeed in investigating them
  - Teachers shouldn’t authoritatively “give” students information, but instead help them discover new ideas
  - Teachers must still provide enough administrative and instructional support to ensure that the class is prepared to tackle the primary goals of each lesson in a timely manner
- Discussion is key, because it allows students to ask the questions that will help them make the necessary linkages between information
  - Teachers should encourage and organize discussion by posing questions that naturally lead to learning in the areas that need to be covered
  - Teachers must still moderate discussion to ensure it stays on-topic and on schedule to meet the course goals
  - Teachers should be as unbiased as possible in their presentation of topics so students can understand and have their own reasoned opinions about them
- Students working in groups provide an excellent vehicle for discussing and carrying out investigations; group work is an essential skill on its own as well

A facilitator will:

- Alert and prepare students for each lesson’s talking points at the beginning of the lesson.
  EXAMPLE: In today’s lesson you will learn about how to calculate the threshold value of a robot using a light sensor… This alerts the student to what you believe is important; calculating the threshold value of the light sensor.

- Ask leading questions rather than just give answers.
  EXAMPLE: If a student asks “How do I calculate how fast is my robot moving?” The teacher may ask, what are the variables that you are related to speed. This can lead into a discussion on how far the robot traveled and how long it took the robot to get there allowing the student to determine the speed themselves.

- Identify the purpose of each lesson and work to keep all students on task and completing the pre-specified lesson.
EXAMPLE: Allow students to ask questions about other sensors used in robotics, but avoid conversations that take the students off track and away from the key focal point of the lesson.

- Prepare for each lesson by creating questioning strategies that scaffold understanding
  EXAMPLE: Asking questions pertaining to a circle’s radius, diameter, and circumference equation will help to ease the transitions when describing that a circle’s circumference is equal to the distance that the robot will travel for one rotation of the robot’s wheel.

- Ask open ended questions that intrigue students to re-examine their existing knowledge.
  EXAMPLE: Students already know that touch sensors can tell the robot what’s in front of it by bumping into objects. Ask if there is anyway for the robot to know what is in front of it without touching it. Ask if there are animals that can detect what’s around them without seeing or touching. This a good starting point for a discussion about sonar and ultrasound.

- Acknowledge time constraints and work to achieve the objectives within them
  EXAMPLE: Limit the number of activities you conduct in any given class based on whether your district uses periods or block scheduling. This helps ensure that any new learning is complete and students aren’t bored by too little work or stressed from too much.

- Create a learning environment that is distraction free and suitable for all students to feel comfortable participating in
  EXAMPLE: Ensure that all students have an equal opportunity to state their opinion or suggest a solution and that no individual’s thoughts are censored or ridiculed by others’

- Reinforce student cooperation and collaboration with the aim of achieving the objective
  EXAMPLE: If one student excels in programming but not in building, pair him/her with someone who is a good builder that way they can learn from each others’ strengths.

- Summarize portions of discussion to emphasize key points and ensure all students grasp what was concluded
  EXAMPLE: During a discussion of Line Tracking, summarize why an outside point on a spinning disc goes faster than an inside point. Make sure all students understand this concept, and then guide the discussion to how this idea can be used to improve the robot’s line tracking abilities.

- Track, recall, and build on long term learning objectives in a manner that is consistent and accessible to students
  EXAMPLE: Create a wall poster with pictures and descriptions of each programming block that they will learn during the class.

- Ensure students create multiple solutions when appropriate and do not accept the first one created out of laziness
EXAMPLE: If students are brainstorming how to make their robot move faster, the most obvious solution would be to increase the motor power. Although this works, if you insisted they come up with more they may discover on their own that changing various drive gears can result in increased axles turns.

- Attempt to gain participation from all students
  EXAMPLE: Ensure that all students have an equal opportunity to state their opinion or suggest a solution and that no individual's thoughts are censored or ridiculed by others'

A facilitator won’t:
- Give the student the answer without making the student work for it
- Allow one or more participants to seize the discussion thus limiting others' participation
- Overuse yes/no questions to control the direction of the discussion
- Support a particular argument because it's consistent with their opinion
- Allow students to intentionally avoid particularly difficult areas of discussion
Introduction to What is a Robot?

Description of the Activity
In this activity students will learn:

• A definition for what a robot is
• How LEGO sensors work

Prerequisites
Student access to the online training materials

Central Concepts
Research
Technological systems
Communications - written and presentation

Approximate Class Time
1 - 2 periods (~45 min each) plus homework.

Note to the Teacher
Robotics and embedded systems make up a multibillion dollar industry that will have the same effect on the emerging economy as computers had on the Information Age. One definition at Carnegie Mellon is S-P-A or Sense, Plan, Act.

Robotic Intelligent Systems (RIS) are programmable electronic embedded systems that are found everywhere. RIS encompass everything from programmable home thermostats to the DARPA Urban Grand Challenge winner, Carnegie Mellon’s “Boss”, a car that drove sixty miles through city traffic without human assistance. RIS are found across all industry sectors. RIS are being integrated into existing technologies to provide flexibility, efficiency, and new feature sets. Robotic technologies and systems are found in phones, cars, homes, offices, hospitals, schools, business, industry, and the military. Robotics is a transformational technology that is embedded into all industries will affect every citizen of the United States.

Class assignment: Conduct research and develop a definition for “What is a Robot”.

The LMS includes a powerpoint that shows examples of industries where robotics will play a central role in the near future.

Students will be able to:
1. Define what a robot is.
2. Describe what sensors are, and their role in a robot’s operation.
3. Describe how LEGO sensors work.
Classroom Implementation

1. Present this chapter’s PowerPoint, “What Is a Robot?”, to the class. You should modify the slides to fit your classroom and teaching style. To keep the discussion real and current for students, consider typing “What is a robot?” into a search engine and seeing what definitions come up - there will be many. Do they have anything in common?

2. Watch several of the Robotics Video links that are included with this chapter in the LMS, but avoid getting caught up in the details at this time. For now, focus on the big ideas about sensing the environment, and the role of the programmed plan in figuring out how to respond correctly. Students will work with each of the sensors later on.

3. Discuss robots that your students are familiar with. They may talk about movie robots. Point out that robotic systems are found everywhere, and review slide 2 on this chapter’s PowerPoint - the field of robotics represents a hundred billion dollar opportunity far broader than just what movies show!

4. If time allows and you have the ability, have students research robots - Carnegie Mellon has many links that show current robotics research at Carnegie Mellon Robotics Institute: http://www.ri.cmu.edu/research_project_view.html?menu_id=261

5. Assign students to watch “How does a robot make decisions?” video and then discuss how a robot thinks with your students.


7. If appropriate, there is a robotics glossary that is part of this lesson. It is up to the teacher if they want to use the glossary in their class.
Introduction to the NXT

Description of the Activity
In this activity students will learn:

• How to provide power to the NXT
• How to turn the NXT on and off
• How to download Firmware
• How to run a program
• How to view sensor values
• How to delete programs from the NXT
• How to change the NXT’s volume level

Prerequisites
NXT sets (unassembled OR pre-assembled into robot), charged batteries, NXT Software loaded on computers, student access to the training materials, a space to test-run the robots once they are programmed.

Central Concepts
Reading and listening for understanding
Technological Literacy
Maintenance of Technological systems
Operating common technology - Multifunction display with files and folders

Approximate Class Time
1 period (~45 minutes) if robots are pre-built
2-4 periods if building robots from scratch

Note to the Teacher
The videos in this section are designed to teach students how to use the NXT. Students should view and follow along with the videos directly at their workstations if possible.

This is the first time students will be working with the hardware and software in your class. You should work out important policies in advance, such as:

• Where should the robots be stored at the end of the day?
• How should extra parts be handled?
• Is the student allowed to take the robot apart?
At your discretion, students who finish early can help other groups construct their robots, work on the next Chapter, or do additional research on the sensors that they have attached to the robot.

**Students will be able to:**
1. Charge and install batteries in the NXT
2. Download Firmware to the NXT
3. Navigate the NXT’s menu system to:
   - Run a program
   - View sensor values
   - Delete programs from the NXT
   - Change the volume of the NXT
   - Turn the NXT off
4. Recall and follow classroom rules regarding the use of robots:
   - Where to store the robot at the end of the day
   - Where to place parts that I find on the floor
   - What should I do if I want to take my robot apart?

**Classroom Implementation**

1. Have the students build their robot if the robot is not already built. This is a very significant time investment (an extra 2-3 class periods), so you should consider having student volunteers build the robots prior to their use in class.
2. (Recommended) Build and attach all sensor attachments. Building these now will save time and hassle later, since the LEGO parts will already be out of storage. If you choose to skip this step, you should also instruct students to skip the “View Sensor Values” part of the Primer, as there will be no sensors attached.
3. Present the following videos and have the students follow along with them:
   - Powering the NXT
   - How to download firmware
   - An NXT Primer - Multimedia Resource Guide
4. Check student’s understanding by administering the Chapter 3 Quiz (linked at the end of the other Chapter 3 documents).
Introduction to Programming Robot

Description of the Activity
In this activity students will learn:

- The role of the programmer
- How to break a robots behaviors into small behaviors
- How robots make decisions
- How to navigate the NXT programming language

Prerequisites
Student access to the online training materials

Central Concepts
Role of the programmer
Communicating with the robot
Methodological decomposition of a task into subtasks

Approximate Class Time
1 period (~45 minutes)

Note to the Teacher
Programmers play a very important role on any robotics team, but students often have trouble characterizing that role more specifically than “program the robot”. In fact, the programmer’s task is all about communicating with the robot – learning to “speak” a programming language, and thinking about problems in an extremely structured and detail-oriented way that makes sense to a computer.

The concepts that are taught in this chapter are Big Ideas and crucial for a student to understand as they move forward as robot programmers. However, students will generally not have the necessary experience or perspective to understand their value initially. Focus on introducing these concepts for now, and try to make connections back to them in the future when solving larger programming problems.
**Students will be able to:**
1. Describe what the role of the programmer is.
2. Describe how to break robot behaviors down into smaller behaviors.
3. Describe how a robot makes a decision.

**Classroom Implementation**

1. Ask students to describe what they think programming is.
2. Assign students to watch the “What is programming?” video and discuss the role of the programmer. Ask the students if they can think of other things that are programmed. They might come up with web-pages, animations in movies, cars, banking, games, manufacturing, energy... The point needs to be made that learning to program is a very important skill to have in today's world.
3. Ask students how to think about the tasks that need to be programmed. How do you talk to a machine? This will lead into the “Planning Robot Behaviors” video, which teaches students to break large robot movements into small behaviors which can be defined programmatically.
4. Review how a robot makes a decision using the “How Does a Robot Make a Decision?” video. This video appeared earlier, but may be appropriate to review at this time.
5. Hand out the strategy to Solve Programming Problems handout. Review the handout with students.
6. Assign students to write in their own words how to program a robot.
Introduction to Moving Straight

Description of the Activity
In this activity students will learn:

- How to navigate the NXT programming software
- How to program the Move block
- How to change the power level, direction, and units of the move block
- How to iteratively test and adjust a program

Requirements
Charged NXT Robots, NXT Software loaded on computers, student access to the training materials, 4’x4’ game board for the challenge, additional space to run the robot if possible.

Central Concepts
- Computer Programming
- Writing and Communications
- Applied Mathematics - Proportionality
- Working in Teams
- Iterative Testing

Approximate Class Time
1-4 periods (~45 minutes each) depending on how strongly the teacher chooses to emphasize the mathematical aspects behind how far the robot travels.

Note to the Teacher
Many students will note that their robots do not move perfectly straight; in fact, no robot does. Sensors can mitigate the effects of “drift” in the future, but for now, students can minimize the inconvenience by ensuring that their robots are built correctly, and that the rear wheel is aligned so that it does not have to flip around when starting a run.

The Introduction to NXT Robot Programming course focuses on teaching students basic programming. However, the topic of distance and movement provides many opportunities to teach highly relevant mathematics topics such as measurement and proportionality.

Diameter and circumference come up very naturally when discussing robot movement. Students control distance in the software by specifying the number of Rotations that the wheel should turn. As the wheel turns, its outer edge “pushes” along the ground, causing movement. The distance “around” the wheel is its circumference, so every time the wheel makes one rotation, the robot is pushed forward one circumference’s worth of distance!

There are also opportunities to introduce proportional relationships: if a robot travels X cm in 1 Rotation, how far will it travel in 2 Rotations or 5 Rotations?
The Moving Forward unit is designed in four parts: Connect, Construct, Contemplate, and Continue. “Connect” makes a connection between a real world application and the lesson students are studying. “Construct” guides students step-by-step through a basic programming concept. “Contemplate” allows students to explore the concept in greater depth. “Continue” is an open ended challenge that students are assigned to solve.

1. **Connect** - Zoe is a Carnegie Mellon research robot that designed for desert exploration.

2. **Construct** - Students will learn how to program the Move Block so that it travels three rotations forward and three rotations backward.

3. **Contemplate** - Students learn how to change power levels, Using Timing as a method of control, use degrees as a method of control, adjust distance, and the difference between braking and coasting.

4. **Continue** - The Close Shave challenge allows students to apply the concepts in a new situation. This challenge should be modified to align with your overall course goals. Students can guess and test different movement settings to demonstrate proficiency with the Move block, or focus on the use of mathematics to avoid the need to guess.

**Students will be able to:**

1. Locate and use the Move Block in the NXT Programming Software
2. Configure the Move block to travel a specific distance.
3. Change the rate that the robot travels by changing the settings on the move block.
4. Change the direction that the robot travels using the move block.
5. Describe how to change power levels, control the robot using timing and degrees, how to adjust distance, and the difference between coasting and braking.
6. Solve a gameboard challenge dealing with movement (Close Shave Challenge).

**Classroom Implementation**

1. Have a robot for every two students which is charged and ready for them to use.
2. Assign student to open the “Moving Straight Unit” in the LMS. The unit is designed for self-paced student use. You may wish to ask students to check in with you upon reaching the last page so that you can present the challenge before they begin.
3. (Optional) Assign students to keep a journal, writing a short paragraph about what they learned in each step of the Contemplate section.
4. Introduce the Close Shave challenge and have the students solve the problem. Consider asking students to document the process that they used to find the settings needed to travel the correct distances.
Moving Straight Challenge
In this challenge, you must drive your robot as close as possible to a LEGO minifig without knocking the minifig over. The minifig will be placed at one of three locations, with the specific spot determined by a coin toss.

Rules and Procedure
1. Load any programs you intend to use onto the NXT. Remember that you can load more than one onto the NXT at a time, and choose which one to run later.
2. Start the robot behind the black line (no parts overhanging).
3. Place the minifig 1 floor tile away from the line, or 30 cm if your floor does not have tiles.
4. Flip two coins.
5. For each “heads” result, move the minifig back one tile (or additional 30 cm increment).
6. Choose a program from the ones you have prepared, and run it.
7. Calculate your score:
   • Start with 100 points.
   • Add 5 bonus points for difficulty for each “heads” you got in the coin toss.
   • Measure the distance from the closest piece on the FRONT of your robot to the FRONT of the minifig’s chest (you can determine which is the front of your robot).
   • Subtract 5 points for every 0.1 cm of distance between the minifig and the robot.
   • Earn a score over 80 points to beat the challenge!
Introduction to Turning

Description of the Activity
In this activity students will learn:

• Two types of turning techniques
• How to describe the robot’s turns in degrees
• How to iteratively test their program

Requirements
Charged NXT Robots, NXT Software loaded on computers, student access to the training materials, 4’x4’ game board for the challenge, additional space to run the robot if possible.

Central Concepts

| Computer programming                  | Writing and Communications |
| Applied mathematics - Proportions     | Teamwork and Problem Solving |
| Direction and Angles                  | Iterative testing           |
| Circles and Degrees                   |                             |

Approximate Class Time
1-4 periods (~45 minutes each) depending on how strongly the teacher chooses to emphasize the mathematical aspects behind the robot’s turning.

Note to the Teacher
The Introduction to NXT Robot Programming course focuses on teaching students basic programming. However, the topic of direction and turning provides many opportunities to teach highly relevant mathematics topics such as angular measurement.

In this unit, students will learn about swing turns and point turns. They will also test to see which turn is better to solve the Maze Challenge.

The “Labyrinth” maze challenge should be modified in to suit your course objectives. Allowing students to guess-and-adjust their turning values will allow students to focus on learning the programming software. Emphasizing the use of mathematics to find the correct rotation values will better align the lesson with mathematics standards. Asking students to present their solutions to their classmates can emphasize technical writing and communication standards.

Some students will finish the challenge before other students. You may wish to allow students to work ahead, or provide additional challenge boards for them to navigate.
The Turning unit is designed in four parts: Connect, Construct, Contemplate, and Continue. “Connect” makes a connection between a real world application and the lesson students are studying. “Construct” guides students step-by-step through a basic programming concept. “Contemplate” allows students to explore the concept in greater depth. “Continue” is an open ended challenge that students are assigned to solve.

1. **Connect** - Nomad is a Carnegie Mellon research robot designed to explore the Arctic.

2. **Construct** - Students will learn how to program the robot to execute two types of turns, a swing turn and a point turn.

3. **Contemplate** - Students learn how to adjust the turns using proportionality and what a 90 degree turn is.

4. **Continue** - The Labyrinth maze challenges students to put together multiple straight movements and turns to complete a program.

**Students will be able to:**
1. Describe the difference between point and swing turns.
2. Program both swing turns and a point turns, in both left and right directions.
3. Program a robot to perform a 90 degree turn.
4. Guess-and-adjust to make their robots turn any number of degrees.
5. (Optional) Use proportional reasoning to program any turn without guessing.
6. Describe what they learned in the Contemplate section.
7. Complete a challenge combining straight and turning movement (Labyrinth maze).

**Classroom Implementation**
1. Have a robot for every two students which is charged and ready for them to use.
2. Assign student to open the “Turning Unit” in the LMS. The unit is designed to allow the student to move self paced through the instructional materials.
3. (Optional) Assign students to keep a journal, writing a short paragraph about what they learned in each step of the Contemplate section.
4. Assign the Labyrinth maze challenge and have the students solve the problem. If applicable, have students describe the process that they used to find the settings needed to make the turns.
Turning Challenge
This challenge features a sequence of turns that the robot must perform in order to get to the end of the maze. The robot must begin at the starting point (#1 below), and get to the goal area (#2) using moving and turning behaviors.

Rules and Procedure
1. Start the robot in the Start area (marker #1)
2. The robot must stop in the Goal area (marker #2).
3. The robot must not cross any lines while traveling (except the “stripes” in the Goal area).
4. Reach the goal to beat the challenge!

Note: Diagrams are not drawn to scale

1 Starting point
2 Goal area
Introduction to The Touch Sensor

Description of the Activity
In this activity students will learn:

• How to program a robot to go forward until it touches an object
• How to program a robot to go forward until it releases an object
• How a touch sensor works

Prerequisites
Charged NXT Robots with Touch+Light Attachment, computers with NXT Programming Software, student access to the training materials, 4’x4’ game board for the challenge, additional space to run the robot if possible.

Central Concepts

<table>
<thead>
<tr>
<th>Touch sensor mechanism</th>
<th>Writing and Communications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Programming: Wait For block</td>
<td>Teamwork and Problem Solving</td>
</tr>
</tbody>
</table>

Approximate Class Time
1 - 2 periods (~45 minutes each).

Note to the Teacher
The Touch Sensor is the first of four sensors that the students will learn about. It detects whether the orange “nub” on the front of the sensor is pressed in or not. When it is pressed in, it is considered “pressed”. When it is not pressed in, it is considered “released”.

Be careful with your use of language! “Pressed” and “Released” refer to the state of the sensor at a particular moment, and not the act of pressing or releasing. If the orange part is held down, it is “pressed” the entire time, not just in the instant it is pushed in.

The Touch Sensor does not have its own attachment in the provided designs. Instead, it shares an attachment with the Light Sensor for space reasons. Pay careful attention to which port each sensor is plugged into.

The “Wait for Touch” block (shown at left) is introduced in this lesson. This is a specific instance of the “Wait For” family of blocks (which happens to wait for a Touch Sensor press/release).

Students will be able to:
1. Describe how a touch sensor works.
2. Write a program that uses a touch sensor to control their robot.
3. Complete a board challenge (Vacuum challenge) requiring the robot to respond appropriately to a Touch Sensor.
This unit is designed in four parts: Connect, Construct, Contemplate, and Continue.

1. Connect - An Autonomous Vacuum cleaner that navigates its world by detecting when it bumps into things. Ask students if they can think of other examples of touch sensors - “Can you think of other places where something responds to a touch?”

2. Construct - Students will learn how to program the touch sensor so that the robot moves forward until the touch sensor makes contact with a wall.

3. Contemplate - Students learn how to write a program so that the robot moves forward until the touch sensor is released, and review how the touch sensor works.

4. Continue - The Vacuum Challenge allows students to build a program that simulates the robotic vacuum cleaner.

Classroom Implementation

1. Have a robot for every two students which is charged and ready for them to use.

2. Assign the students to go through the “Touch Sensor Unit”.

3. (Optional) Assign students to keep a journal, writing a short paragraph about what they learned in each step of the Contemplate section.

4. Have students watch the “How does a touch sensor work?” video and write a description of how a touch sensor works.

5. Assign the Vacuum challenge. If applicable, have students describe the process that they used to choose the blocks needed to make create the behaviors they used.
Move until Touch Challenge
In this challenge, you must program your vacuum-equipped robot to touch all four walls of a rectangular room, regardless of the room’s size.

Rules and Procedure
1. Create a “room” with any dimensions, as long as the walls form a rectangle.
2. Keep the room empty (no furniture or other obstructions).
3. Position your robot any way you wish at the start.
4. Run your program – you must use the same program regardless of the size of the room.
5. Your robot must make physical contact with all four walls of the rectangle during a single run (without getting stuck) to complete the challenge!
Teacher Notes - The Ultrasonic Sensor

Introduction to The Ultrasonic Sensor

Description of the Activity
In this activity students will learn:

• How to program a robot to stop at a certain distance from an object without touching it
• What a threshold value is
• How an ultrasonic sensor works

Requirements
Charged NXT Robot, NXT Programming Software, Ultrasonic Sensor attachment, student access to the training materials, 4’x4’ game board for the challenge, additional space to run the robot if possible.

Central Concepts
How ultrasonic sonar sensors work
Threshold values
Programming: Wait For block

Approximate Class Time
1 - 2 periods (~45 minutes each).

Note to the Teacher
The Ultrasonic Sensor uses sonar to measure distance. It emits a sound wave, measures how long it takes for the sound to bounce off an object, then multiplies by the speed of sound to determine what the distance must have been (distance = speed * time). “Sonar sensor” and “ultrasonic sensor” are used interchangeably in the NXT context.

Some Ultrasonic sensors “see” slightly farther to the sides and will detect parts of the robot’s own body (especially the Touch+Light Sensor) as an obstacle. If this happens, move the sensor higher and/or farther forward so it can see over the rest of the robot.

The “Wait for Near” block (shown at left) is introduced in this lesson. It is a “Wait For” block, like the Wait for Touch block.

The Threshold divides the many possible numeric values that the sensor can measure (0 - 255 cm) into two categories: Near and Far (below or above the threshold, respectively). The Wait For block then simply Waits for Near or Waits for Far.

The challenge for this unit is very similar to the Turning Unit’s Labyrinth maze challenge, but with standing walls. If you still have the Labyrinth board available, you may wish to reuse it as a foundation for this challenge. Any tall, hard object works as a wall. Use classroom materials such as books, or even the boxes the LEGO parts came in.
This unit is designed in four parts: Connect, Construct, Contemplate, and Continue.

1. **Connect** - An Autonomous tractor that is being developed at Carnegie Mellon. Many leading agricultural systems manufacturers are conducting significant research into integrating robotic systems into their existing vehicles.

2. **Construct** - Students will learn how to program the sonar sensor so that the robot stops when it comes within a certain distance of an object.

3. **Contemplate** - Students learn how write a program so that the robot moves backward when the robot sees an object that is too close. The will also review how the sonar sensor works and what a threshold value is.

4. **Continue** - The Walled Maze challenge allows students to write a program that can find its way through a walled maze.

**Students will be able to:**
1. Describe how an ultrasonic sensor works.
2. Write a program that uses an ultrasonic sensor to control their robot.
3. Describe what a threshold is.
4. Complete a board challenge (Walled Maze challenge) requiring the robot to respond appropriately to an Ultrasonic Sensor.

**Classroom Implementation**
1. Have a robot for every two students which is charged and ready for them to use.
2. Assign student to go through the “Ultrasonic Sensor Unit” in the LMS.
3. (Optional) Assign students to keep a journal, writing a short paragraph about what they learned in each step of the Contemplate section.
4. Assign the students to watch the “Introduction to the Sonar Sensor”
5. Assign students to complete the Walled Maze Challenge.
Move until Near: Walled Maze Challenge

This challenge features a slightly varied layout using standing walls. The robot must begin at the starting point (#1 below), and get to the goal area (#2) using moving and turning behaviors. However, the dimensions of the hallways may change, and your robot must use its sensors to help navigate the maze!

Rules and Procedure
1. Start the robot in the Start area (marker #1)
2. The robot must stop in the Goal area (marker #2).
3. All walls should be constructed to be solid and elevated (use wooden planks or books).
4. Wall lengths may vary, but turns will always remain the same.
5. Reach the goal to beat the challenge!

Note: Diagrams are not drawn to scale

1. Starting point
2. Goal area
Introduction to The Light Sensor

Description of the Activity
In this activity students will learn:

• How to find the threshold value using the view menu
• How to program the robot to travel and stop at a dark surface
• How to program the robot to travel and stop at a light surface
• How a light sensor works

Requirements
Charged NXT Robot, NXT Programming Software, Touch+Light Sensor attachment, student access to the training materials, 4’x4’ game board for the challenge, additional space to run the robot if possible.

Central Concepts
How light sensors work      Threshold Values
Programming: Wait For block  Writing and Communications
Teamwork and Problem Solving

Approximate Class Time
2 - 3 periods (~45 minutes).

Note to the Teacher
The Light Sensor shines a red light onto a nearby surface and measures how much of the light reflects back. Brighter or redder surfaces will reflect more red light and give a higher reading. Darker or less-red surfaces reflect less and produce a lower reading. Light Sensor readings do not have a specific unit like lumens. They are only 0-100 from dark to light.

The Light Sensor provides critical navigation capabilities, but its readings fluctuate based on environmental factors. Try to keep shadows and lighting as consistent as possible.

If problems arise, first ask students to recheck their thresholds by using the View Mode. Very often, this will solve the problem.

The “Wait for Dark” block (shown at left) is introduced in this lesson. It is a “Wait For” block, like the Wait for Touch block.

The Threshold divides the many possible numeric values that the sensor can measure (0 - 100) into two categories: Dark and Light (below or above the threshold, respectively). The Wait For block then simply Waits for Dark or Waits for Near.

A convenient formula for estimating a good threshold is (use View Mode):
(Value over Light surface + Value over Dark surface) / 2

© 2011 Carnegie Mellon Robotics Academy
This unit is designed in four parts: Connect, Construct, Contemplate, and Continue.

1. **Connect** - “Boss”, Carnegie Mellon’s entry into the DARPA Grand Challenge. This robot autonomously navigated 60 miles through a city traffic, obeying all traffic signals and speed limits.

2. **Construct** - Students will learn how to calculate threshold values, stop when the robot’s sensor sees a dark line, stop when the sensor sees a light surface, and how the light sensor works.

3. **Contemplate** - Students learn how to program the robot to move forward until it sees a light surface, and how to calculate a threshold value.

4. **Continue** - The Intersections challenge allows students to write a program that can drive around a block, looking for painted stop lines in a way similar to the Boss robot.

**Students will be able to:**
1. Describe how a light sensor works.

2. Write a program that uses a light sensor to control the robot.

3. Calculate a working threshold value for a light sensor.

4. Complete a board challenge (Intersections challenge) requiring the robot to respond appropriately to a Light Sensor.

**Classroom Implementation**

1. Have a robot for every two students which is charged and ready for them to use.

2. Assign students to go through the “Light Sensor Unit” in the LMS.

3. (Optional) Assign students to keep a journal, writing a short paragraph about what they learned in each step of the Contemplate section.

4. Have students watch “How does a light sensor work?” and discuss why the sensor is so prone to interference from external light sources.

5. Review the procedure and reasons to calculate the threshold value for a light sensor, as this is tricky and must be done frequently.

6. Assign students to complete the Intersections Challenge.
Move until Touch Challenge

In this challenge, you must program your robotic vehicle to drive around a model city block, stopping appropriately at each intersection according to the stop lines, then proceeding.

Rules and Procedure

1. Mark off a rectangular area to represent the block. The block may have any dimensions, but must always be a rectangle.
2. Leave the roads clear of other vehicles and obstacles.
3. You may position your robot in any way you wish at the start.
4. Your robot must drive around the block, stopping at each stop line for at least one second.
5. Your robot must NOT stop in other places.
6. You may use only one program regardless of the rectangle’s size.
7. Complete one full lap around the block to beat the challenge!
Introduction to Repeating Behaviors - Introduction to Loops

Description of the Activity
In this activity students will learn:

• How to repeat behaviors using the NXT programming language
• How to control loops using counting and sensing

Requirements
Charged NXT Robot, NXT Programming Software, Touch+Light Sensor attachment, Ultrasonic Sensor attachment, student access to the training materials, 4’x4’ game board for the challenge, additional space to run the robot if possible.

Central Concepts
How loops work                                  Writing and Communications
Programming: Loops                              Teamwork and Problem Solving

Approximate Class Time
3 - 4 periods (~45 minutes each)

Note to the Teacher
Loops are used to repeat behaviors. In this unit students will learn to use a loop block. Unlike simpler blocks, the Loop Block has an “inside” area (where the two Move Blocks are shown, below) as well as an “end cap” block on the right end of the structure.

The loop repeats the “inside” blocks over and over until a condition set in the “end cap” block is met. When the “end cap” condition is met, the loop will end.

There is an important nuance to this, however. Students often believe that the Loop block will automatically terminate and move on when the end cap condition is met, regardless of what the program is doing at the time. This is incorrect; the end cap condition is ONLY checked when the program has finished all of the “inside” blocks (i.e. when the “end cap” is reached). If the blocks inside the loop take 10 seconds to finish, the robot is only able to check the end cap condition once every 10 seconds!

Often, students understand the idea of looping, but not the idea of how the loop determines when to STOP looping. Make sure they understand both before moving on.
This unit is designed in four parts: Connect, Construct, Contemplate, and Continue.

1. **Connect** - FANUC Assembly line robot. Robots are used around the world to increase productivity and accuracy at repetitive tasks.

2. **Construct** - Students will learn how to write a program that repeats its behavior using a Loop Block.

3. **Contemplate** - Students learn how control the number of times the loop repeats using counting and feedback from sensors. Students will also begin to realize that programs are not always linear now that loops can cause groups of blocks to run more than once – review the loop animation on slide 8 of the slide show.

4. **Continue** - The Automated Sentry challenge allows students to write a program that controls a robot that travels around a building.

**Students will be able to:**
1. Describe how a loop works
2. Write a program that uses a looping behavior
3. Complete a board challenge (Automated Sentry challenge) requiring the robot to repeat a sequence of actions for a set number of times.

**Classroom Implementation**

1. Have a robot for every two students which is charged and ready for them to use.
2. Assign students to go through the “Repeating Behaviors Unit” in the LMS.
3. (Optional) Assign students to keep a journal, writing a short paragraph about what they learned in each step of the Contemplate section.
4. Print out “Strategies to Solve Programming Challenges” for students. Explain that they are now building more complex programming structures, and it will become more important to keep perspective on what problems (and sub-problems) they’re solving. Have students begin to describe their programs in words before they write them, breaking tasks into smaller behaviors whenever possible.
5. Assign the “Automated Sentry” Challenge. Make it clear in advance that their programs are not expected to work perfectly at this stage - they will discover that Loops alone will NOT solve all the problems (this transitions into the next unit on Switches)
Repeating Behaviors Challenge
Your company has been commissioned to design a robot to patrol outside a building. You are tasked to develop a “proof of concept” prototype that can travel a designated “loop” route.

Rules and Procedure
1. Build the board below, including the central building and surrounding fence.
   - You don't need a solid object for the building, just the cornerposts are sufficient.
   - The building should be rectangular in shape.
   - Posts are 1” diameter PVC tubing or cardboard paper towel tubes.
2. Your robot must complete two full laps around the building.
3. Your robot must not come into contact with any of the posts marking the building or fence.
4. Your program must incorporate at least one Loop Block.
5. Complete two laps using a loop to beat the challenge!

Note: Diagrams are not drawn to scale
Introduction to Obstacle Detection

Description of the Activity
In this activity students will learn:

• How to use the Switch Block
• How to create and use a Switch Loop – a conditional statement embedded in a loop
• How to program an obstacle detection behavior

Requirements
Charged NXT Robot, NXT Programming Software, Touch+Light Sensor attachment, Ultrasonic Sensor attachment, student access to the training materials, 4’x4’ game board for the challenge, additional space to run the robot if possible. Loops unit completed.

Central Concepts
Programming: Conditional Statements           Writing and Communications
Programming: Looped Conditional Statements    Teamwork and Problem Solving

Approximate Class Time
3-5 periods (~45 minutes each), as this program requires frequent troubleshooting

Note to the Teacher
This lesson teaches two very important concepts back-to-back. Conditional statements, called Switches in the NXT Programming Software, let the robot choose between two different actions based on a sensor value or other condition.

Equally important, however, is the pattern of embedding a Switch into a Loop. This vital programming technique allows the robot to repeatedly check a sensor or other condition. Because the robot processes instructions so quickly, the rapidly-repeated decision looks “continuous”. Students use this technique to make the robot “continuously” watch for obstacles while moving.

It may seem strange that the Switch Block alone has so few robotics applications, but it is very rare that the robot has to make a decision only once.

Consider this idea from a Sense-Plan-Act perspective. The robot must constantly be checking its sensor values in order to react appropriately – it cannot just check once and rely on stale information forever!
This unit is designed in four parts: Connect, Construct, Contemplate, and Continue.

1. **Connect** - The Cyberguard Security Robot “continuously” watches for intruders while making its patrol rounds.

2. **Construct** - Students will write a program that chooses to stop or go based on an Ultrasonic Sensor Switch block. Then they embed the Switch Block in a Loop, creating a Switch-Loop.

3. **Contemplate** - Students will study how the Switch Block works. Slide 8 shows the program flow, a visual illustration of where the “active” part of the program is, and how its path is decided by sensor values at switches and loop end caps. Have students study this carefully. Students will then modify their program’s Switch-Loop so that the robot stays a specific distance from an object.

4. **Continue** - The Automated Sentry 2 challenge builds on the Automated Sentry challenge from the Repeating Behaviors unit by allowing the robot to “continuously” monitor for obstacles while moving.

**Students will be able to:**
1. Explain how a Switch and a Switch-Loop work
2. Write a program using a Switch-Loop to move and watch for obstacles at the same time
3. Complete a board challenge (Automated Sentry 2 challenge) requiring the robot to move and monitor for obstacles at the same time.

**Classroom Implementation**

1. Have a robot for every two students which is charged and ready for them to use.
2. Assign students to go through the “Obstacle Detection Unit” in the LMS.
3. (Optional) Assign students to keep a journal, writing a short paragraph about what they learned in each step of the Contemplate section.
4. Discuss Slides 8 and 9 in the Contemplate section, making sure students understand how Switches and Switch-Loops control which blocks run, and when.
5. Review “Strategies to Solve Programming Challenges” again. Remind students that these complex programming structures will be more manageable if they can keep perspective on what problems (and sub-problems) they’re solving.
6. Assign students to complete the “Automated Sentry 2” Challenge.
7. Even if students are not completely clear on Switch-Loop concepts, they can still move on to the next chapter, Line Tracking, which primarily deals with the same concept.
Obstacle Detection Challenge
Your company’s prototype passed the first phase of testing! For the second phase, your robot must demonstrate the ability to detect and prevent collisions with people and objects along its route.

Rules and Procedure
1. Build the board below, including the central building and surrounding fence.
   • You don’t need a solid object for the building, just the cornerposts are sufficient.
   • The building should be rectangular in shape.
   • Posts are 1” diameter PVC tubing or cardboard paper towel tubes.
   • Obstacles can be anything you like, but must be detectable by the Ultrasonic Sensor.

2. Your robot must complete two full laps around the building.

3. Your robot must not come into contact with any of the posts marking the building or fence.

4. Your robot must not come into contact with any of the obstacles!
   • Obstacles may be placed anywhere, at any time.
   • Your robot must wait for obstacles to be removed before continuing along its path.

5. Complete two laps while encountering at least four obstacles to beat the challenge!

6. HINT 1: The rotation sensor “accumulates” rotations between loops and makes “until rotations” hard to use in this situation... the black lines may offer a useful alternative.

7. HINT 2: You don’t need to detect obstacles while turning; the angle of the sensor isn’t very good at it anyway.

Note: Diagrams are not drawn to scale
Introduction to Line Following

Description of the Activity
In this activity students will learn:

• How to program a line following behavior
• How to control the duration of the behavior using timing or sensors
• How to program a robot to track the other side of the line
• How to tune parts of the behavior to perform better under known conditions

Requirements
Charged NXT Robot, NXT Programming Software, Touch+Light Sensor attachment, student access to the training materials, 4’x4’ game board for the challenge, additional space to run the robot if possible. Loops and Obstacle Detection units completed.

Central Concepts
Emergent Behaviors Writing and Communications
Programming: Looped Conditional Statements Teamwork and Problem Solving

Approximate Class Time
1 - 4 periods (~45 minutes each), depending on how well students understood Switch-Loops from the previous chapter

Note to the Teacher
Because Switch-Loops are so critical but also so complex, this lesson deals with the same programming structure as the previous one, now in a new context. The emphasis in this lesson is mastery rather than introduction – the different line tracking behaviors will require students to manipulate different parts of the structure to achieve the desired results.

The Line Tracking behavior is “emergent” in the sense that it is a very simple decision (forward-left vs. forward-right movement) that produces the relatively complex behavior of line following.

Review Light Sensor threshold calculations and use – the Light Sensor will be used intensively in this activity, and an incorrect threshold or carelessly cast shadow will cause great confusion.
This unit is designed in four parts: Connect, Construct, Contemplate, and Continue.

1. **Connect** - Materials Transport Robot. The AMTS Robot project at Carnegie Mellon involved a Tug robot autonomously transporting parts from place to place in a factory. This technology has evolved significantly and automated transports are now common in factories and warehouses.

2. **Construct** - Students will learn how to write a program that repeatedly chooses between forward-left or forward-right turns to track a line.

3. **Contemplate** - Students learn how to control when the line-tracking behavior ends, and how to optimize the behavior to be more efficient on certain types of lines. Students will also learn how to make the robot track the other side of the line.

4. **Continue** - The RoboSlalom challenge is timed, so students will need to string together multiple line-following behaviors that are optimized for straight vs. sharply curved sections of the course.

**Students will be able to:**
1. Describe how robot is able to track a line with just a single repeated decision
2. Program a robot to track a line for a specific distance
3. Complete a timed board challenge (Roboslalom) requiring multiple, differently-tuned variations on the basic line-following behavior.

**Classroom Implementation**

1. Have a robot for every two students which is charged and ready for them to use.
2. Assign students to go through the “Line Following Unit” in the LMS.
3. (Optional) Assign students to keep a journal, writing a short paragraph about what they learned in each step of the Contemplate section.
4. Discuss the other implications of Slide 8, which describes how a Line Tracking behavior can be limited to run for only a certain amount of time. How else could it be limited?
5. Discuss slide 9 in the Contemplate section, making sure students understand the trade-offs that the robot makes between fast-but-straight power settings and slow-but-sharp ones, and that these changes “optimize” the robot’s behavior for different environments.
6. Review “Strategies to Solve Programming Challenges” again. In this lesson, focus on the fact that the slalom must be broken into several differently-optimized line following behaviors in order to complete the course within the time limit.
7. Assign students to complete the “Roboslalom” Challenge.
Line Following: Slalom Challenge

Line Following Challenge
In this challenge, your robot must reach the end of the slalom course as quickly as possible, without knocking over any of the posts along the route.

Rules and Procedure

1. Build the board below.
   - The line marking the course should be dark tape on a light-colored surface, though you can use light tape on a dark surface if needed.
   - Posts should be 1” PVC or paper towel tubes.
   - You can modify the shape of the curves to make them sharper or more gradual.
   - The curves do not need to be the same. Some can be sharper than others. There can be straight areas between gates or not. Be creative!

2. Use a stopwatch to time your robot’s run from the time the Run button is pressed, to the time it crosses the finish line.
   - Knocking over any gatepost incurs a 3-second penalty per post.
   - Missing a gate (not going between the posts) incurs a 10-second penalty per gate.

3. Complete the course in under 15 seconds (including penalties) to beat the challenge!

Hints
- You can run different optimized line tracking behaviors one after another to go fast or turn sharply when needed.
- How might you limit them so each one stops at the correct time?

Note: Diagrams are not drawn to scale.
Introduction to Line Following

Description of the Activity
In this activity students will learn:

• How to use multiple sensors and behaviors to solve a complex programming challenge
• Concepts of iterative software development, including spreading tasks across phases and frequent testing
• How to solve a large, complex problem by breaking it down

Requirements
Charged NXT Robot, NXT Programming Software, Touch+Light Sensor attachment, Ultrasonic Sensor attachment, student access to the training materials, 4’x4’ game board for the challenge (multiple setups if possible, for testing). All prior course units completed.

Central Concepts
Engineering: Iterative Development  Teamwork and Problem Solving
Engineering: Project management

Approximate Class Time
5+ periods (~45 minutes each). You should pick a deadline and ask students to plan around it, as part of the Project Management aspect of this challenge.

Note to the Teacher
Important: Slides 6-8 of the Obstacle Course sequence provide a guided walkthrough of the solution. Instruct students NOT to view this section if you want them to work through the problem on their own!

The Obstacle Course Grand Challenge serves as both the capstone project for the course, and also an opportunity to introduce engineering project management skills and techniques which only make sense in the context of relatively complex projects like this one.

Slide 5, Iterative Design, is the focal point of the new concepts for this chapter. Students usually try to solve too many parts of the problem at once, and quickly run into failure and frustration. A step-by-step iterative method provides a practical and effective alternative.

Troubleshooting Tools
There are two mini-lessons included in this chapter: Play Sound and Display to Screen. These two lessons explain how to insert audible or visible troubleshooting cues into programs. They can be made mandatory, optional, or skipped at the instructor’s discretion.
Students will be able to:
1. Select and program appropriate behaviors to solve a complex gameboard challenge
2. Manage problem-solving and solution-development processes effectively using an iterative development model

Classroom Implementation

1. Have a robot for every two students which is charged and ready for them to use.
2. (Optional) If time allows, assign students to watch the “Play Sound” and “Display to Screen” videos. Discuss to make sure that students understand when these tools could be useful.
3. Assign students to view Slides 1-2 of the “Obstacle Course Challenge”. Hand out the Obstacle Course Challenge handout and discuss the challenge with students.
4. Have students view Slide 3 of the “Obstacle Course Challenge”. Discuss the breaking down of the problem into sub-problems, following the same reasoning as “Strategies to Solve Programming Challenges” from before.
5. Watch and discuss slide 5 in “Obstacle Course Challenge”. Discuss the two methods that don’t tend to work, and why. Ask students to use their own words to describe what it means to iteratively build their solutions to this challenge. Ask what their first steps might be for the Obstacle Course Challenge.
6. (Optional) Assign a daily journal entry where students log their attempts (both successes and failures) for each day.
7. Let students begin solving the Obstacle Course Challenge.
8. Monitor students’ progress, watching for common pitfalls like attempting to add too many behaviors in a single step.
9. Remind students of the project’s deadline. If teams seem unlikely to complete the challenge, point out that it worth more points to get some steps working well than to have a whole program full of broken behaviors.
10. When the pre-set deadline arrives, collect student programs and journals.
Obstacle Course Challenge
This challenge features a sequence of tasks that the robot must perform in order to get to the end of the course. The robot must move from its starting area to touch the wall, follow a line, get to a safe area, then move toward a final goal zone. A combination of both previously-learned and new behaviors will be needed to accomplish these tasks.

Rules and Procedure
1. Start the robot in the Start area (#1)
2. The robot must touch the NXT storage box (#2).
3. The robot must not knock down the pipe obstacles or the book while following the line (#3).
4. Once the robot is inside the safe zone (#4), it can be picked up and moved, but you cannot start a new program.
5. The robot must finish in the final goal (#5).
6. Complete all numbered steps in a single run to beat this challenge!

Note: Diagrams are not drawn to scale