

2

the NXT-G programming environment

This chapter takes a close look at the NXT-G programming environment and presents a few simple programs. The NXT-G programming environment is fairly complex, with lots of features. This chapter starts with the basics, and later chapters cover some of the more advanced features. All the programs in this chapter use only the NXT Intelligent Brick.

a tour through the MINDSTORMS environment

When you start the MINDSTORMS application, you will see the Getting Started window, shown in Figure 2-1. From this window you can either start a new program or open an existing one.

NOTE The screenshots in this book are from the NXT 2.0 retail kit version of the software running on Windows XP. They may appear slightly different on your screen if you use a different version or operating system.

Click the Go button in the Create New Program section to create a new empty



Figure 2-1: The Getting Started window

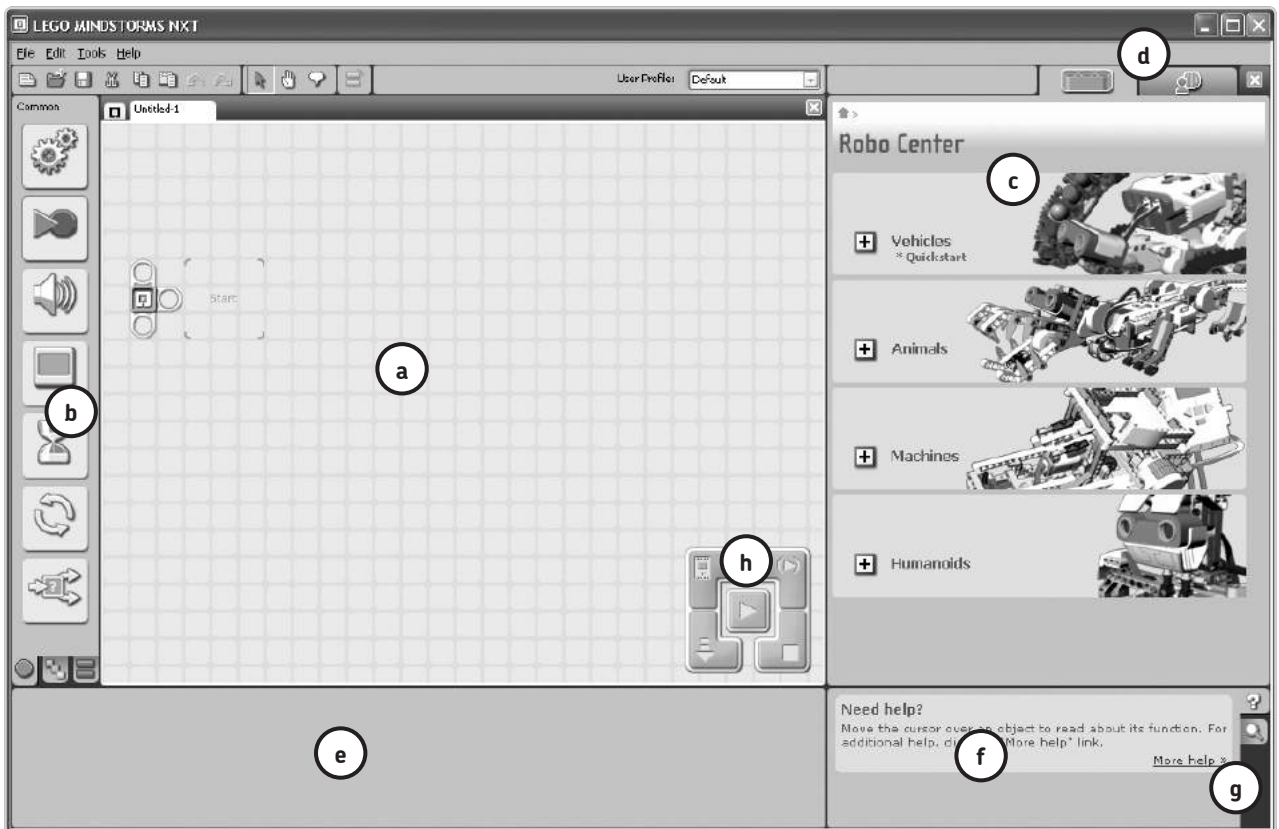


Figure 2-2: The MINDSTORMS NXT environment

and untitled program, as shown in Figure 2-2. Before writing your first program, let's look at the sections of the MINDSTORMS IDE.

a: work area

In the center of the screen is the Work Area, which is also known as the *Program Sheet* because it resembles a sheet of graph paper. Tabs across the top of the Work Area let you select between your open programs. The tab on the left with the little square selects the Getting Started window that is displayed when you first start the IDE.

b: programming palettes

To the left of the Work Area are the Programming Palettes, which contain all the blocks used to create programs. To help keep things simple, there are three palettes, selected using the tabs at the bottom, as shown in Figure 2-3.

The first tab selects the *Common Palette*. This group has the most commonly



Figure 2-3: The Programming Palette tabs

used blocks, giving you quick access to the blocks that you'll use most frequently. The center tab selects the *Complete Palette*. All the available blocks, including the ones on the Common Palette, appear on the Complete Palette. The final tab selects the *Custom Palette*. Blocks that you create, called *My Blocks*, will appear on the Custom Palette.

c: robo center

The area to the right of the Work Area can hold either the Robo Center window or the My Portal window (the education set versions use the name *Robot Educator* instead of *Robo Center*). This area provides building and programming instructions for some example projects, with each version of the software providing different projects. I encourage you to work through these projects; they are a great way to become familiar with the MINDSTORMS environment. Closing this area gives you a larger Work Area.

d: my portal window

As mentioned, the area to the right of the Work Area can hold either the Robo Center window or the My Portal window. The

My Portal window shows links to interesting areas on the LEGO MINDSTORMS website. The education set versions of the software connect to the LEGO Education site. When you select the My Portal tab, this window replaces the Robo Center window and should appear something like the following:



e: configuration panel

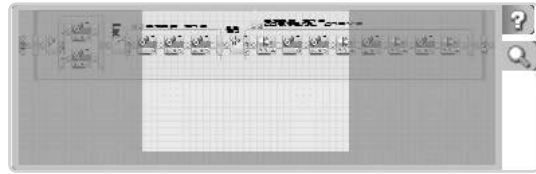
At the bottom left of the screen is the *Configuration Panel*. Use this area to configure the blocks that make up your program.

f: help panel

At the bottom right of the screen are the Help and Navigation Panels. The *Help Panel* shows a short description of the currently selected block. Click the More Help link to open the full help file, which contains a complete description of every block. Select the Help Panel by clicking the Question Mark tab on the right.

g: navigation panel

When working with large programs, the *Navigation Panel* lets you select which part of your program to display in the Work Area. Select the Navigation Panel by clicking the Magnifying Glass tab. This replaces the Help Panel and displays the entire program, so you cannot see the details of each block, but you can tell which part of the program is displayed in the Work Area. The Navigation Panel for a large program may look like the following:



h: controller

In the bottom-right corner of the Work Area is a group of five buttons called the *Controller*. The Controller is the connection between the programming environment and the NXT Intelligent Brick.

writing an NXT-G program

Writing an NXT-G program is a fairly straightforward process. In the Work Area is what looks like a small white LEGO beam, which is called the *Sequence Beam*. You create an NXT-G program by dragging blocks from the Programming Palettes onto the Sequence Beam. Specify the exact behavior of each block using the Configuration Panel. When you run the program, the NXT executes each block in the order they appear on the Sequence Beam. The blocks are run one at a time, meaning that each block must finish its operation before the next block is started. The program ends when the end of the Sequence Beam is reached. The order of the blocks and the way they are configured determines how the program behaves.

The previous description is actually a little simplistic. A few blocks do not necessarily complete before the next block starts, for example. You can also use more than one Sequence Beam, which complicates how the blocks are run and when the program ends.

NXT-G is a very convenient language for humans, but the NXT needs something a little different. Your NXT-G program is called the *source* or *source code*, and it needs to be translated into a set of instructions that the firmware knows how to execute. The process of translating a program from a human-friendly language to one used directly by the computer is called *compiling* the program.

Once you write a program, use the Controller to compile and download the program to the NXT. *Downloading* is the process of copying the program and any other files the program needs from your computer to the NXT.

your first program

For your first program, use the Sound block to have the NXT say “Hello.” When you start the MINDSTORMS IDE, you’ll see an area that lets you create a new program or open an existing one. In the Create New Program box, enter **Hello** as the program name, and click the **Go** button, as shown in Figure 2-4. (This box is labeled *Start new program* in the original NXT retail kit and the education set.)



Figure 2-4: Creating a new program

A new Work Area will open. Follow these steps to add a Sound block to the new program:

1. Select the Sound block from the Common Palette, as shown here:
2. Drag the block onto the end of the Sequence Beam, which looks like a little white bar. Place the Sound block right on top of the area conveniently labeled *Start*, as shown here:



Your program should now look like the following:



If you accidentally grab the wrong block or drop it in the wrong place, select **Edit ▶ Undo** on the menu and start again.

The Configuration Panel for the block should be displayed below the Work Area. If you do not see it, click the block, and the Configuration Panel should appear.

You’ll leave most of the items on the Configuration Panel at their default values and change only the Sound File setting. Select **Hello** in the list of available sound files, as shown in Figure 2-5.



Figure 2-5: The Sound block’s Configuration Panel

saving your work

Before continuing, save your program by selecting **File ▶ Save**. When you first save a program, a dialog will let you name the program file and select the location to save it, as shown in Figure 2-6.

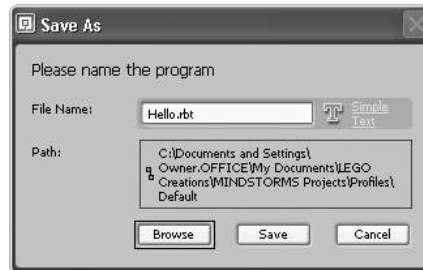


Figure 2-6: Saving your program

The filename defaults to *Hello.rbt* because you entered *Hello* as the program name when creating the new program. If you neglected that step, the name will be something like *Untitled-1*, but you can change it to something more meaningful.

When you click the **Save** button, the file *Hello.rbt* is created. This file contains all the information about your program, including the blocks you used and how they are configured and arranged. The file format is unique to the MINDSTORMS environment; you won’t be able to edit it using other programs.

NOTE Save your work often. Save before downloading your program and certainly before getting up to answer the phone or walk the dog. Redoing a few hours of work because you neglected to save your program is really annoying.

MAKING BACKUP COPIES

Most of the time when you are working on a program, you'll make progress. However, every now and then when you change your program, instead of making it better, you end up with a horrible mess, and you can't seem to get back to what you started with. Professional software developers use fancy tools called *source code control systems* to save versions of their work to avoid this problem, but you can get the same benefits by copying working versions of your program to a different folder. Use one of these backup copies if you run into trouble. It's a good idea to save a copy after getting each new feature working and before making large changes. For example, if you are working on a program with four tasks, you might save it as *Task1* after you get the first part working. When you start adding the second task, you might save it as *Task1_Task2*. This way you could always go back a step if necessary.

running your program

Once you save your program, it's time to try it. The first step is to make sure the NXT is turned on and connected to your computer using either a USB cable or a Bluetooth connection. The USB connection is easier to set up (just plug in the cable between the NXT and your computer) but more limiting; a Bluetooth connection can be more difficult to set up but gives you more freedom. If you have problems getting a Bluetooth connection to work, try looking at the NXT hardware forum at <http://www.NXTasy.com/> for several useful tips.

Click the center Play button in the Controller (shown in Figure 2-7) to download and run your program.


When you click the Play button, a dialog should appear with the messages Compiling, Downloading, and Complete. Once the messages finish, your NXT should respond by saying "Hello."



Figure 2-7:
The Controller

your second program

The next program, *HelloDisplay*, is very similar to the *Hello* program, except that instead of using the Sound block, you'll use the Display block to write *Hello* to the NXT display. One big difference between this program and the first one is that this program won't work the first time. This will give you a chance to look at what to do when your program doesn't work. Use the following steps to create the initial version of the program:

1. Create a new program called *HelloDisplay*. Open the Getting Started window by clicking the small tab on the top of the Work Area that looks like this: 
2. It will be the first tab on the left, before any tabs for programs you have open.
3. Fill in the program name in the Create New Program box, and click the **Go** button as shown earlier in Figure 2-4.
4. Drag the Display block from the Common Palette onto the Sequence Beam. Your program should look like this:



5. The block's Configuration Panel should be displayed, as shown in Figure 2-8. Click the block to select it if the Configuration Panel is not showing.

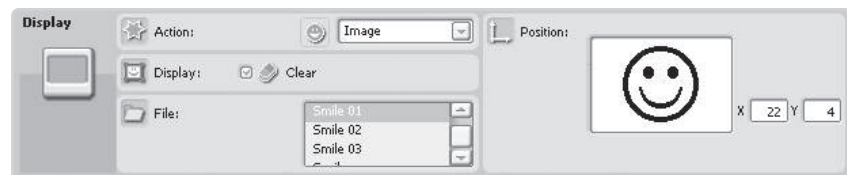


Figure 2-8: The Display block Configuration Panel

- To display the word *Hello*, you need to change the Action value from Image to **Text**. Click the arrow beside the word *Image*, and select **Text** from the pop-up box:



- Changing the Action value causes the Configuration Panel to display a different set of options. It should now look like Figure 2-9.



Figure 2-9: The Configuration Panel after setting the Action value to Text

- The last step is to replace *Mindstorms NXT* with *Hello* in the Text box. Figure 2-10 shows the completed settings with the changes highlighted.

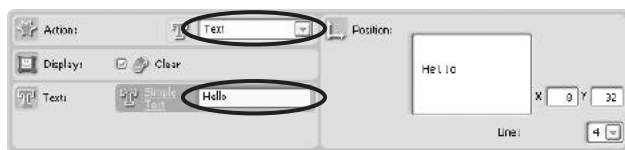


Figure 2-10: Final Display block configuration

Now download and run your program. The NXT will beep twice to let you know it has downloaded a program, but *Hello* will not show up on the display. If you closely watch the display while you run the program, you will see that the NXT starts the program and then immediately says that it's done.

debugging

What happened? To put it simply, this program has a bug. A *bug* is a program error. *Debugging* is the process of finding and fixing errors. Like every programmer, you'll spend a lot of your programming time debugging. Fixing a bug can sometimes be a frustrating process, but it can also be incredibly rewarding to track down and solve a difficult problem. Think of it as solving a puzzle, and remember that you should always have fun!

Of course, some puzzles are easier to solve than others. For example, most people would agree that a Rubik's Cube is a fairly complex puzzle, but even it can be solved by following a simple set of rules (a quick Internet search for *Rubik's Cube Solver* will lead you to a list of solutions).

Like a Rubik's Cube, debugging a program isn't always easy, but it helps to follow a set of rules. Although there is no definitive set of steps for finding a bug, if you follow some simple techniques, you'll find it much easier to debug your programs.

reproduce the bug

First see whether you can reproduce the bug, which just means seeing whether you can get the program to repeat the same behavior. For example, if you try running this program again, it will fail again and will fail the same way every time you run it. This is actually a good thing. A bug that's difficult to reproduce is also more difficult to fix. (Later chapters deal with some trickier ones, but for now we have a reasonably well-behaved bug.)

simplify the program

Normally, the second step is to simplify the program as much as possible, but since this program is already as simple as it can be, there is no way to do so. If it were larger, you could remove pieces to concentrate just on the area with the problem. Another approach is to try to reproduce the same problem using a small test program.

look at the parts of the program

Next, examine the parts of the program to try to figure out what is happening. This program has only two parts—the Display block and the Sequence Beam—so it's likely that one of these is not working quite the way you want.

Think about the difference between how you expect the program to behave and the observed behavior. The program consists of the single Display block, which should write *Hello* to the NXT's display screen, but instead the NXT shows that the program is finished. You know the program ends when all the blocks on the Sequence Beam have been run. A reasonable explanation is that the program ends before you have a chance to read the display because there are no other blocks on the Sequence Beam.

fix the bug

Now that you have a possible explanation for the bug, you can quickly test a possible solution (in most cases, finding the cause of a bug is much more difficult than solving the problem).

the edit-compile-test cycle

You can attempt to fix the program by adding a Wait Time block after the Display block. Use the Wait Time block to tell the program to pause for five seconds before ending, which will give you enough time to read the display.

The Wait Time block does not appear directly on the Common Palette; it is in a group of blocks that wait for different conditions. Hover the mouse cursor over the hour glass icon to make all these blocks appear. Figure 2-11 shows these blocks, with the Wait Time block circled.



Figure 2-11: The Wait Time block on the Common Palette

Use the following steps to fix the program:

1. Drag a Wait Time block onto the Beam to the right of the Display block. Your program should look like Figure 2-12.

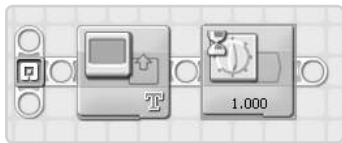


Figure 2-12: The program with the Wait Time block added

2. By default this block pauses for one second. To have a little more time to read the display, change the Until value from 1 to 5. This will make the program wait five seconds before ending. Figure 2-13 shows the Configuration Panel with the change.



Figure 2-13: Waiting for five seconds

Now when you download and run the program, the display should show *Hello* for five seconds before clearing, which is the behavior you want. Adding the Wait Time block is a successful solution to this bug.

The process you just went through is an example of the typical programming process. In fact, it even has a name: the *edit-compile-test cycle*.

It is very unusual for any program to work the first time. After writing your program, download it to the NXT and test it. Testing will often reveal some problem, so you return to editing. Just keep going through the cycle, adding features and fixing problems. Eventually, all the features are added and all the problems are worked out to give you a complete working program. It can take some patience and perseverance, but it is a great feeling when it finally works.

NOTE Why didn't the first program have the same problem? It's because the Sound block has a Wait for Completion option that is selected by default (see Figure 2-5). This makes the program wait until the sound plays before continuing. If you uncheck the Wait for Completion option in the first program, it will fail in the same way.

comments

Programmers use comments to add descriptive text to their programs. Every programming language that I am aware of allows programmers to do this, and NXT-G is no exception. Use comments to tell other programmers how your program works or why you made certain decisions. For example, you could add a comment to the previous program to explain why you added the Wait Time block.

In the previous chapter, I mentioned that a good program should be easy to modify and understandable to other programmers. Good comments are important in achieving both of these goals. It can be very difficult to figure out how a program works just by looking at the settings for each block. A short description in plain English (or whatever language you speak) will make your program much easier to understand. Think of how you might describe your program to a friend. You wouldn't just list the blocks you use; instead,

you'd describe what the program does as a whole, and you might also explain how the more complicated parts of your program work. Use comments to add this type of information to your program so it will be much more useful to other programmers. Comments also help you remember why you wrote a program in a particular way, making it easier to reuse your own programs.

The arrangement and configuration of blocks on the Sequence Beam is all the information that the NXT needs to run your program. Using the previous program as an example, however, a person will want to know *why* you added the Wait Time block. On the other hand, you just need to tell the NXT to wait five seconds. The reason is unimportant. Comments do not affect how a program runs; the NXT will completely ignore them.

adding comments

In this section you will add two comments to the HelloDisplay program. The first comment will describe what the program does, and the second will explain the reason you added the Wait Time block.

the program description

A Configuration Panel exists specifically to hold a description of the program. A reasonable description of this program is *Say "Hello" using the display*. Entering the program description is a simple two-step process:

1. Click the MINDSTORMS icon on the left side of the Sequence Beam (shown here) to open the Configuration Panel for the program description.



2. Enter the description **Say "Hello" using the display** in the box provided in the Configuration Panel, as shown here:



the comment tool

The second comment will be placed before the Wait Time block to explain why that block is there. A reasonable explanation is *Wait 5 seconds before ending the program to give the user time to read the display*.

It may be a little easier to understand the process of adding a comment if you can see the final result. Figure 2-14 shows the program with the comment added. Anyone looking at this program will know why the Wait Time block is there.

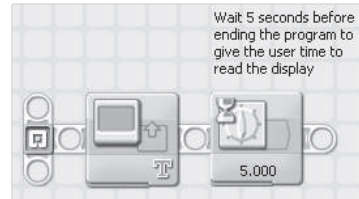


Figure 2-14: Explaining the Wait Time block

Add comments to your program using the Comment tool on the toolbar, as shown here:



Use the following steps for adding this comment:

1. Click the **Comment** tool on the toolbar. When you move your mouse over the program, you should see the I-beam cursor usually used when working with text.



NOTE The mouse cursor may look a little different if your system settings are not the same as mine.

2. Click above the Wait Time block. A small black box will appear on the screen to let you know where the comment will be placed, as shown here:



3. Start typing the comment **Wait 5 seconds before ending the program to give the user time to read the display**. Press the ENTER key to move to the next line when the text gets wider than the block (it's easier to read if the comment does not extend way past the block).

4. When you finish adding a comment, you can start another one by clicking in the Work Area. When you finish adding comments, select the Pointer tool from the toolbar.
5. This tells the IDE that you want to go back to selecting blocks. When you move your mouse cursor over the program, you should now see the normal pointer, as shown here:



rules for working with comments

The following are a few things you should know about comments:

- * Pressing ENTER while typing a comment makes it go to the next line.
- * Clicking the text of a comment lets you change the comment.
- * Double-clicking in the Work Area starts a comment.
- * Clicking the edge of a comment selects the comment.
- * You can delete the selected comment by pressing the DELETE key.
- * You can move a selected comment by dragging it with the mouse; just make sure to click the edge of the comment and not the text.

the configuration panel

Before moving on to the next program, let's take a closer look at the Configuration Panel. When you select a block in your program, the Configuration Panel for that block displays in the lower-left corner of the IDE. This is where you set the options that control exactly how the block behaves. You will spend a lot of time working with the Configuration Panel when writing an NXT-G program.

Each block has its own Configuration Panel, and subsequent chapters discuss the details of each block. In this section, I concentrate on how the Configuration Panel works in general. The National Instruments engineers did a good job of using a consistent look and feel across the Configuration Panels for all the blocks, which is one of the features that make NXT-G easy to use.

I'll use the Configuration Panel for the Sound block as an example for this discussion. This is a fairly typical block, and all the ideas discussed here are repeated in many other blocks.

general layout

Each block has a different set of items you can configure, but those items tend to be arranged in the same way. Figure 2-15 shows the Configuration Panel for the Sound block. This is how the Configuration Panel looks when you first add the Sound block to a program, before you make any changes.



Figure 2-15: The Configuration Panel for the Sound block

The items are arranged in two columns. The items you set first will be in the left column, with the most important ones near the top. For example, you can use the Sound block to play either a sound file or a single musical tone. This is controlled by the Action item, located at the top of the first column. The Action is listed first because the type of sound affects how some other items behave.

changing panels

The Sound block has many options, and some of the choices can only be used together. In the Hello program, you left the Action item set to Sound File and selected Hello from the Files list on the right. However, it only makes sense to select a sound file from the list when you select Sound File for the Action. When you select Tone, you need to make different choices. Figure 2-16 shows the Configuration Panel with the Tone action selected. Compare this with Figure 2-15. Notice that the right side of the panel has changed. The list of files has been replaced with items used to select a note to play.



Figure 2-16: Configuration Panel for playing a single tone

Many blocks have Configuration Panels that change like this, where an item on the left side controls the choices that appear on the right. This allows a single block to have several different uses without displaying a confusing list of conflicting options.

disabled items

Occasionally, you may find that the option you want to select is disabled. A disabled item appears on the Configuration Panel but is light gray and you cannot select it. This happens when some other incompatible choice is selected. For example, the Sound block can play a sound or stop a sound that is currently playing. When you select Stop for the Control item, the rest of the choices become disabled, as shown in Figure 2-17. This makes sense because there is no point in setting the volume or selecting a note to play when you want the sound to stop.



Figure 2-17: With Stop selected, the rest of the choices are disabled.

a block's configuration icons

You can select a block and examine its Configuration Panel to see exactly what it does. However, you can look at the Configuration Panel for only one block at a time. Fortunately, the way a block is displayed in the program changes based on some of the most important configuration items. Here is the way the Sound block looks in the Hello program:



Notice the three icons along the bottom. These icons change based on your configuration choices. The icon on the left tells you that the action is set to Sound File. The same

icon appears next to the Sound File choice on the Configuration Panel. The middle icon shows that you are playing a sound, and the third icon shows the Volume setting.

This is how the Sound block appears when you select Tone for the Action setting:



The icon on the left has changed to a small musical note to match the Action setting.

When the Control item is set to Stop, the icons for the Action and Volume settings are not shown; only the Stop icon appears:



Each block uses a different set of icons because the blocks all have different settings. If you are not sure what an icon means, look at the Configuration Panel for the block. The icons shown on the block will also appear next to one of the settings.

conclusion

This concludes our tour of the MINDSTORMS environment. I covered the basics that you will need to create simple NXT-G programs, and you'll get plenty of practice as we progress. I will introduce more advance features as we need them.

The next step is to build a simple TriBot to use with the example programs in the following chapters. Then I'll introduce the many blocks available in NXT-G and show you how to put them together to make the TriBot perform a variety of tasks.