LEGO® MINDSTORMS® NXT-G Programming Guide

James Floyd Kelly

Apress®

LEGO® MINDSTORMS® NXT-G Programming Guide

ISBN-13 (pbk): 978-1-59059-871-9

ISBN-10 (pbk): 1-59059-871-7

Printed and bound in the United States of America 9 8 7 6 5 4 3 2 1

Lead Editors: Jim Sumser/Jonathan Hassell
Technical Reviewer: Fay Rhodes
Editorial Board: Steve Anglin, Ewan Buckingham, Gary Cornell, Jonathan Gennick, Jason Gilmore, Jonathan Hassell, Chris Mills, Matthew Moodie, Jeffrey Pepper, Ben Renow-Clarke, Dominic Shakeshaft, Matt Wade, Tom Welsh
Project Manager: Richard Dal Porto
Copy Edit Manager: Nicole Flores
Copy Editor: Heather Lang
Assistant Production Director: Kari Brooks-Copony
Senior Production Editor: Laura Cheu
Compositor: Ellie Fountain
Proofreader: April Eddy
Indexer: Broccoli Information Management
Artist: Kinetic Publishing Services, LLC
Cover Designer: Kurt Krames
Manufacturing Director: Tom Debolski

Distributed to the book trade worldwide by Springer-Verlag New York, Inc., 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax 201-348-4505, e-mail orders-ny@springer-sbm.com, or visit http://www.springeronline.com.

For information on translations, please contact Apress directly at 2855 Telegraph Avenue, Suite 600, Berkeley, CA 94705. Phone 510-549-5930, fax 510-549-5939, e-mail info@apress.com, or visit http://www.apress.com.

The source code for this book is available to readers at http://www.apress.com in the Source Code/Download section.

For Dan, Belle, Sarah, and Jake—my "Texas family"

Contents at a Glance

Contents

About the Author

JAMES (JIM) FLOYD KELLY is a freelance technical writer and currently lives in Atlanta, Georgia. With degrees in English and Industrial Engineering, his friends and family often wondered what he was thinking about when he made that decision. Well, he somehow managed to turn his skills into a career where he gets to play with robots, new software, and other technologies. Jim was one of the original Mindstorms Developer Program (MDP) participants selected by LEGO to test the new Mindstorms NXT robotics kit, and he contributes, with other NXT experts, to The NXT Step Blog (`http://www.thenxtstep.com`). He is also a member of the Mindstorms Community Partners (MCP), a group of NXT testers that continues to work with LEGO on the NXT product line.

About the Technical Reviewer

FAY RHODES is a freelance graphic designer with a love of learning. She's become a great fan of LEGO Mindstorms NXT and is currently the only member of The NXT Step Blog who is a woman. Fay has gifts for building things and problem solving, which are put to good use designing creative, noncompetitive NXT robots.

Acknowledgments

I always read the acknowledgments page when I buy a new book. I like to know a little more about who helped shape a book and how it came to be, so I hope you'll take a second and read a little about the excellent team I got to work with to make the book that you're holding a reality.

A huge *thank you* goes to the team at Apress. You can read a complete list of the names of the persons involved in this book a few pages back, but just know that everyone at Apress has worked hard to bring this book to completion. I do need to point out the three individuals that I was most involved with during the editing of this book—Richard Dal Porto, Laura Cheu, and Heather Lang. The book is much improved as a result of their hard work.

Technical books need another pair of eyes to check and double-check that errors are eliminated. For that job, I get to thank my technical editor, Fay Rhodes (and her husband, Rick). I've worked with Fay and Rick on The NXT Step Blog for some time now, and it was nice to have them checking my work. This book is better because of their due diligence.

And finally, I have to acknowledge my fellow contributors at The NXT Step Blog for their ideas, questions, concerns, comments, and general sharing of information. Many times I approached them with questions of my own and got straight and accurate answers. For everything from screenshots to the CAD building instructions to great programming examples, I must thank them all.

Introduction

So, you want to learn to write really great programs for your Mindstorms NXT robots, huh? I totally understand. You can build the most awesome robot you and your friends have ever seen, but if all it does is spin around or count from one to three, no one is going to be impressed. What you want is to match the amazing construction of your bot with an amazing program, right? Absolutely! You wouldn't be reading this book if you weren't interested in improving your programming skills.

Let me first tell you that I'm not a programming guru. I've had many computer programming classes in my past, but I don't do this for a living. What I do is enjoy building and programming my own little collection of bots, just figuring out things as I go. I will be the first to admit that many of the programs I write are ugly (meaning they are sometimes quickly thrown together without any planning involved). But some programs I write are extremely elegant (the opposite of ugly—they're well planned, developed, and tested). But whether the program is ugly or elegant, it usually works. It works, because I spend the necessary time figuring out how to use the tools that are provided with the NXT programming language.

And that's the key point I want you to take from this Introduction: *The more familiar you are with the tools available to you in the programming language, the more easily you'll be able to write some powerful programs for your robots.*

This book isn't about the techniques for building robots, although I'm going to give you some building instructions—you can download the instructions for free by going to `http://www.apress.com` and clicking Source Code/Download under Quick Links. This book *is* about programming your bots.

Finally, if you've got any questions or comments about the book, feel free to e-mail me. My e-mail address is `jktechwriter@gmail.com`, and I'd love to hear from you.

James Floyd Kelly
Atlanta, GA
June 2007

CHAPTER 1

Robots and Programs

If you are already familiar with the subject of robots and the concept of programming, feel free to skip ahead a few chapters. But if you are just starting out with your LEGO Mindstorms NXT robotics kit and are asking yourself questions such as, "How is a robot different from a toaster?" or "Just what is this thing called programming?" then you're in the right place. If terms like "conditional statements," "nested loops," and "variables" make your head spin, don't worry—they make my head spin, too.

There is simply no reason that learning to use the Mindstorms NXT robotics kit should cause stress. It's supposed to be fun, right? Building robots and making them do what you want them to do shouldn't cause headaches. I don't like headaches, and I certainly don't want to give you one, so sit back and let me show you a less stressful method for getting the most out of NXT.

What Is a Robot?

I'm going to keep this short—I promise. What is a robot? There are numerous definitions. One definition is a human-shaped mechanical device that mimics human actions. Another definition is an electronic machine that functions independently, without human control. And there are many more. There truly doesn't seem to be one official definition.

For the purpose of this book, I'm going to give you my definition. Here goes: *A robot is a device that is built to independently perform actions and interact with its surroundings.*

In a nutshell, a robot should be able to move and react all on its own. If you are controlling its actions, it's just a remote-controlled toy, right? But if your device can do things like examine its surroundings, respond to obstacles such as chairs or walls, pick out a red ball from a mix of colored balls, and hundreds of other activities without help from its human creator, then you've got a robot.

You can build a robot using all the great Mindstorms NXT components that came with your robotics kit. Your bot can have claws or hands. It can have ears to listen and eyes to see. It can walk on legs or roll on wheels. But in order for a robot to be able to do all these things on its own, you must provide it with one additional component, a program.

What Is a Program?

I know I told you that computer terminology makes my head spin, but there are some terms that cannot be avoided. But the terms I want to introduce to you are easy to explain and even easier to spell, so they can't be all that bad!

When we talk about the Mindstorms NXT robotics kit, we're talking about a piece of technology. Technology almost always requires a little learning, but that shouldn't mean it has to be boring—NXT robots are cool and fun. So, let's start right off by defining one of the coolest technical terms you need to understand—*program*.

I can't really write a book about NXT programming without defining what a program is, can I? So let's jump in with a small discussion about this word. I promise to keep it fun.

Let's take a look at a very basic robot. I call this robot SPOT and, for right now, SPOT only does one thing. He sits.

Take a look at Figure 1-1; there's SPOT doing what he does best.

Figure 1-1. *My bot SPOT*

Can we all agree that SPOT is a fairly boring robot? We all know that robots should do things! You could almost say that SPOT needs to be trained. And that's how I'm going to define the word "program." Read the next two sentences slowly: A *program* is a set of instructions for my robot. *Programming* is what you do when you create a program.

It's not a long definition, and it certainly isn't complicated. The definition will get a little more detailed as you read more chapters, but for now, let's just start out with that very basic idea.

You've encountered a lot of programs in your lifetime. Don't believe me? Okay, let me give you an example:

Teacher: Okay, class, take out your history books.

[*Grumbling, the students take out their books.*]

Teacher: I want everyone to turn to page 55.

[*With more grumbling, everyone turns to page 55.*]

Teacher: Everyone read through to page 65.

[*Loud grumbling*]

The teacher just gave a program to follow: take out your book, turn to a specific page, and read a specified number of pages. Let me give you one more example:

Step 1: Place the widget firmly against the whatsit.

Step 2: Snap the special wonder-whatchamacallit into the widget.

Step 3: Flip the whatsit over, and bend the thingamajig to the left.

Those are steps I found in an instruction manual—a program for me to follow. If I follow the steps, my whatsit should work perfectly (my whatchamacallit still isn't working!).

A simple program is just a set of instructions (written, spoken, or maybe provided in some other method) that needs to be followed. I certainly don't want to call you a robot, but in a way, we all can frequently act like robots. When we follow a set of instructions, we are running a program! (Another word you might sometimes hear used instead of "run" is "execute": "I told SPOT to *run* his `SLEEP` program" is the same as "I told SPOT to *execute* his `SLEEP` program.")

Let's go back to SPOT. He's just sitting there. How boring. Let's pretend for a moment that SPOT has ears, and I can give him some instructions. I'll start off by giving SPOT some basic movements:

Me: SPOT, move forward.

[*SPOT starts to roll forward.*]

Me: SPOT, stop.

[*SPOT stops rolling.*]

I've just given SPOT two very simple programs to follow. What? Two programs? Yes, the first program is "Move forward." The second program is "Stop." The simplest programs can be just one step! Now, I could combine them into one program, but I'll encounter a problem:

Me: SPOT, move forward and stop.

[*SPOT just sits there.*]

What happened? Well, think about someone telling you to "move forward and stop." How far forward will you move? When will you stop? You're smart, but robots are not. Robots must be told *exactly* what to do. And in this example, SPOT did exactly as he was told. SPOT moved forward and stopped. The reason you didn't see him move is because the moment he started spinning his motors, he stopped.

In the first example, I waited until SPOT began to roll before telling him to stop, so he had time to actually move. In the second example, I combined the instructions into one program

(move forward and stop) without telling SPOT how far or maybe how long (in time) to move forward. So let's try it again:

Me: SPOT, move forward for 5 seconds and stop.

[*SPOT moves forward for 5 seconds and then stops.*]

Okay, so maybe SPOT isn't the problem. I've just figured out that when I tell SPOT to do things, I've got to be *very* specific.

One other thing that SPOT is good at is reading my handwriting. Let me give you another example of how specific I need to be when telling SPOT to execute a program, but this time, instead of telling him what to do, I simply take out a piece of paper and write down the following: SPOT, move forward 3 inches; turn left 90 degrees; move backward 2 inches; spin 360 degrees, and stop.

Next, I give the piece of paper to SPOT, and he reads it. He moves forward 3 inches, turns left 90 degrees, moves backward 2 inches, spins 360 degrees, and, finally, stops.

If your NXT robot is like mine, though, it probably doesn't have the ability to listen to voice commands or read a sheet of paper.

If your robot can't hear you or read your handwriting, how exactly do you tell it what to do? Easy! You're going to use programming software. There are other names such as programming suite or graphical programming environment, or blah blah blah—for now, let's just use programming software, OK?

You're in luck—your Mindstorms NXT robotics kit comes with programming software called NXT-G (the *G* is for "Graphical," meaning programs are not written instructions such as my previous handwritten steps for SPOT).

■ **Note** There are a lot of ways to program. Just as different people speak different languages, robots (and computers and other technical stuff) can speak different languages. Some examples of human languages are English, Spanish, French, German, and Italian. For your NXT robots, there are a variety of languages, too. I speak English, because that is the language I learned to speak in school. Your NXT Brick comes from the factory understanding one language: NXT-G.

I also speak Spanish. But it's not my native language. Your NXT Brick can learn to speak other languages, too, but its native language is NXT-G. Most people won't learn another language until they understand their native language well. And that's what you need to do—learn NXT-G well so you can talk to your robot (by giving it a program).

NXT-G

NXT-G is the tool you will use to tell your robots what to do. NXT-G allows you to create programs that can be *uploaded* (installed) to your NXT robot. These programs can be instructions as simple as "move forward 2 inches and stop" or as advanced as you can imagine! NXT robots can be built with a variety of motors and sensors. But without a good program, your

robot won't know what to do: Do I spin my motors? What do I do with this Touch sensor? Without programming, you'll have one confused robot on your hands.

NXT-G is installed on a computer (there are Windows and Macintosh versions) and exists as software. I'm not going to be covering the basics of using the software, so you'll need to refer to the *LEGO Mindstorms NXT User Guide* that came with your NXT kit for installation instructions and steps on how to perform basic steps such as creating new programs, saving programs, and other items.

You will create and save your programs (just like you save a drawing or an essay on your computer) and then connect your NXT robot to the computer. When your NXT robot is connected, you will be able to upload one or more programs to your robot and run (execute) them.

Right now, there are two versions of the Mindstorms NXT-G programming tool. One version comes with the NXT robotics kit that's bought in a store (sometimes called the retail version), and the other comes with NXT kits bought through the LEGO Education division (sometimes called the education version). Education versions are typically purchased by schools and teachers, but anyone can actually buy this version if they wish.

If you're not sure which version you have, take a look at Figures 1-2 and 1-3. Figure 1-2 shows the retail version (with the Robo Center), and Figure 1-3 shows the education version (with Robo Educator). Don't stress too much over the version you own; there are differences, but for the purposes of this book, over 90 percent of the tools are identical.

Figure 1-2. *NXT-G retail version comes with Robo Center*

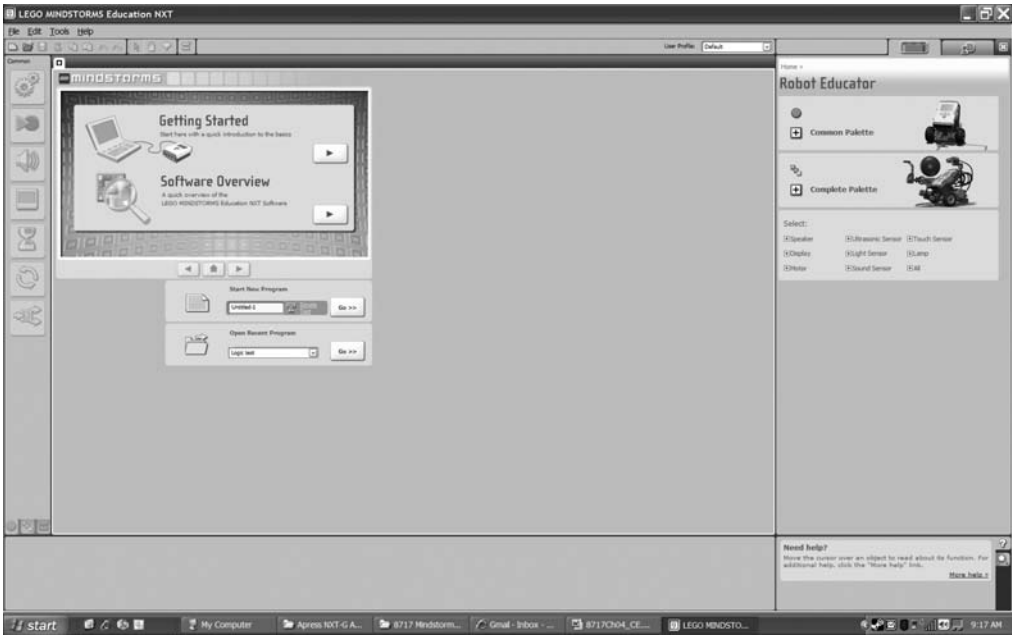

Figure 1-3. *NXT-G education version comes with Robo Educator*

The NXT-G Programming Software is fun to use; feel free play around with it. The best part about NXT-G is that much of it is extremely easy to figure out on your own. When you're ready to start learning how to create some awesome programs, turn to Chapter 2. I'm keeping the chapters short so you'll have plenty of time to read a little and then go play—no 50-page chapters in this book!

The next chapter is going to help you figure out what you want your robot to do. Go and experiment a little with NXT-G, and I'll see you in Chapter 2.

CHAPTER 2

Program Structure

I don't really like using technical terms like "program structure," but it is a very useful concept that will benefit you as you begin to program your robots. So bear with me for this short chapter.

What Do I Mean by Structure?

Back in Chapter 1, I gave you some examples of real-world programs. Would the following example have made any sense?

Teacher: Class, open your books to page 55.

[*The class looks confused.*]

Teacher: Class, I want you to get out your history books.

[*Giving the teacher confused looks, the students get out their books.*]

How can you read page 55 if you haven't yet been told which book to open? You might answer, "Yes, but I'm in history class, and the teacher said turn to page 55. So I'm sure the teacher means my history book!"

That's true. As a human, you are able to figure out certain instructions on your own. But remember—robots aren't that smart! They need to be given very strict and specific instructions. And those instructions need to be given in a specific order. That order is another way of saying "program structure."

Let's get out SPOT for another example. He's still doing his one and only trick—sit. We're not quite ready to upload an NXT-G program yet, but let's do some preplanning at this stage. I want you to use something that computer programmers call *pseudo-code*. What is pseudo-code? Well, the definition of "pseudo" is fake (as in pretend, simulated, virtual—get the idea?); it's not real. And "code" is simply another word for program. So put it all together and one way of looking at pseudo-code is this: fake program.

Our fake program isn't going to be written using NXT-G. The best way I can tell you to start creating a fake program is to pretend that SPOT has ears and tell SPOT what you want him to do. Let's try writing some pseudo-code using a numbered list:

1. SPOT, move forward until your Touch sensor is pressed and released; then stop.
2. Okay, SPOT, I want you to turn left 90 degrees.