Welcome to the world of robotics software! If you are just entering the world of robotics, a warning is in order. Whether you are a hobbyist, a student, or a professional, working with robots can be addictive, and if you are not careful, it can crowd out other worthy pursuits such as your career, social life, and personal hygiene. If you are willing to take that risk, keep reading.

This book is not going to tell you how to build your own robot to fetch beer from the refrigerator or pick up your roommate’s socks off the floor. Instead, you will learn how to use an innovative new software development kit (SDK) called Microsoft Robotics Developer Studio (MRDS) to build software for robots already on the market, as well as custom robots and robots yet to be built.

The best feature of MRDS is that it’s free for personal, academic, and development use. If you end up shipping an application that uses MRDS, there is a modest licensing fee.

If you have a LEGO NXT or an iRobot Create, or one of several other robots, you can start controlling your robot with MRDS right away. If you have a robot that doesn’t yet have MRDS support, this book will show you how to write your own custom robotics services.

You don’t have a robot yet? Don’t worry about it. MRDS includes a sophisticated 3D simulation environment for virtual robots. You can be driving robots through a virtual environment just minutes after you download the free SDK, and the majority of the examples in the book are designed so that they will work with the simulator.

Don’t be deceived by the “price,” though. MRDS includes a sophisticated run-time environment that makes it easy to write powerful asynchronous and distributed applications. Several large corporations are using this environment in non-robotics applications to handle robust serving of web pages and financial transactions. In its short life, MRDS has already eaten up the road in the DARPA Urban Challenge and swum underwater in a robotic submarine.
If we’re ever going to get to the point where robots overthrow our way of life and threaten the very existence of humanity, we all need to get busy writing robotics software. It’s time to download Microsoft Robotics Developer Studio and become a part of the robotics revolution.

**Microsoft Does Robots?**

Most people are surprised to hear that Microsoft has a robotics team and that they are actively working on software for robots. This section describes how this team came about and what Microsoft’s goal is in this area of software development.

It was in late 2003 that robotics started showing up on the Microsoft radar. Tandy Trower, who was working directly for Bill Gates at the time, began to notice that a number of robotics companies and universities with robotics programs were contacting Microsoft. He wrote a “think week” paper for Bill exploring the possibility of developing a product in this space. At the same time, Bill visited several universities and saw that each had at least one robotics project or program that it was eager to show. It was after noticing a lot of activity and a nearly worldwide enthusiasm for this technology that he directed Tandy to conduct a five-month study to determine what Microsoft should do. Tandy determined that there was a business opportunity for Microsoft and recommended that he be the one to form a team to pursue it. The Microsoft robotics team was born (see the Foreword by Tandy Trower at the front of the book for more details).

Tandy drew developers from many parts of Microsoft with various areas of expertise. They worked together to integrate and develop a number of new technologies to support robotics development. Version 1.0 of the Microsoft Robotics Studio SDK was released in December of 2006, and version 1.5 followed in July of 2007, with a refresh in December 2007 that was used as the basis for this book.

According to Tandy, Microsoft sees parallels between the state of the robotics industry now and the state of the personal computer industry 25 years ago. At that time, software written for one computer would not necessarily work with another computer; as a result, there was a very limited market for software development. With the introduction of the IBM PC, computer hardware became more capable and standardized. DOS, and later Windows, served as a software development environment that enabled programs written on one machine to work on all machines.

Microsoft Robotics Developer Studio is an attempt to do something similar with the robotics industry. It provides a software platform and development environment that enable software written for one robot to also work with another robot with similar capabilities. Microsoft hopes that this development environment will help the robotics industry move forward by helping companies to invest more in robot behavior and algorithms than has been possible in the past.

This is an appropriate point to talk about versions and names. In April 2008, right in the middle of the final stages of preparing this book, Microsoft released a CTP (Community Technology Preview) of V2.0 and renamed Microsoft Robotics Studio as Microsoft Robotics Developer Studio. This is the type of event that causes authors to wake up in a cold sweat at night. After much discussion, it was decided to change references in the book from MSRS to MRDS, except where they clearly had to remain the same — i.e., the main installation directory for V1.5 is C:\Microsoft Robotics Studio (1.5). It is hoped that this is not too confusing.

The code on the book’s website will be updated as soon as the final version of 2.0 ships. There is no point in chasing CTPs because they are moving targets. In the meantime, MSRS — sorry, MRDS — has been
designed to allow multiple versions to run side-by-side, so you can continue to work with V1.5 even while you are using a trial CTP of 2.0.

Microsoft Robotics Developer Studio

Components

The Microsoft Robotics Developer Studio SDK consists of a number of components. The Concurrency and Coordination Runtime (CCR) and Decentralized Software Services (DSS) comprise the run-time environment. They are both managed libraries, so the robotics services that operate within their environments are also implemented using managed code. The Visual Simulation Environment is a 3D simulator with full physics simulation that can be used to prototype new algorithms or robots. The Visual Programming Language (VPL) is a graphical programming environment that can be used to implement robotics services. In addition to all of these components, the MRDS team has implemented numerous samples and complete applications to provide programming examples and building blocks for user applications. The following sections provide a brief overview of each of these components, and they are covered in more detail in later chapters.

Concurrency and Coordination Runtime (CCR)

The CCR is a managed library that provides classes and methods to help with concurrency, coordination, and failure handling. The CCR makes it possible to write segments of code that operate independently. It communicates when necessary by passing messages. When a message is received, it is placed in a queue, called a port, until it can be processed by the receiver.

Each code segment can run concurrently and asynchronously, and there is often no need to synchronize them because of the message queues. When it is necessary to wait until two or more operations have completed, the CCR library provides the necessary constructs.

The CCR provides one or more dispatchers, which determine what segment of code is currently running. The total number of threads is usually set according to the number of independent processors on the system, and code segments are scheduled when a thread becomes available.

One element of asynchronous programming that is often overlooked is error handling. It is possible to use exception handling to isolate failures within a single code segment, but the data that caused the error was likely passed to the segment via a message from another executing segment and the exception handler code has no way of knowing where the operation originated or where the error should be reported.

To solve this problem, the CCR provides causalities. A causality holds a reference to a port that is used to report errors. The causality is associated with a message and it follows that message as it is passed to another code segment and then again to another segment. At any time, if an error occurs, a message is passed to the error port associated with the causality. In this way, errors are reported back to the segment of code that initiated the operation.

A complete description of the CCR as well as example code can be found in Chapter 2. You need a good understanding of the CCR, so be sure to work through Chapter 2 carefully.
Decentralized Software Services (DSS)

The CCR enables segments of code to pass messages and run in parallel within a single process. The Decentralized Software Services (DSS) library extends this concept across processes and even across machines.

An application built with DSS consists of multiple independent services running in parallel. Each service has a state associated with it and certain types of messages that it receives called operations. When a service receives a message, it may change its state and then send additional messages and notifications to other services.

The state of a service can be retrieved programmatically by sending a Get message to the service or it can be retrieved and displayed using a web browser. Services may subscribe to be notified when the state of a service changes or when other events occur. Services may also partner with other services so that they can send messages to those services and receive responses.

It requires a different mindset to write software using CCR and DSS. You must think in terms of independent chunks of code that handle messages asynchronously, rather than function calls and threads. Chapters 3 and 4 provide additional details about the DSS library, including code examples, and are important chapters for your understanding of MRDS.

Visual Simulation Environment

Microsoft Robotics Developer Studio includes a full-featured 3D simulation environment complete with physics simulation. A scene from Simulation Tutorial 5 is shown in Figure 1-1.
Chapter 1: Exploring Microsoft Robotics Developer Studio

The simulation environment supports both indoor and outdoor scenes and comes with a variety of simulated robots. It is also extensible so that you can add custom robots, such as the CoroBot in Chapter 6 and the Lynx 6 robotic arm in Chapter 8, and other objects such as walls, furniture, and so on. It has a built-in editor so that you can add and move objects as the simulator is running.

The simulator is useful for prototyping new robotic algorithms prior to running them on the actual robot hardware. It is much cheaper to destroy a virtual robot with a programming error than a real one. In some cases, the robot you want to program may not even exist yet. In this case, the simulator can provide a good testing ground for a model of the robot.

The simulation environment contains models of several robots, including a LEGO NXT, an iRobot Create, a MobileRobots Pioneer 3DX, and a KUKA LBR3 robotic arm. The models for these robots are shown in Figure 1-2.

![Figure 1-2](image)

A number of other simulation entities such as cameras, sky, ground, and various other objects are also provided.

The simulation environment is covered in Chapters 5 through 9.

**Visual Programming Language (VPL)**

In an MRDS application, there are usually several services running. Some of these are low-level services that interface directly with hardware. Others might be simulation services that connect directly to the simulation engine. Usually one or more top-level services control the behavior of one or more robots. These top-level services typically interface only with other services and are called orchestration services. Sometimes it is possible to implement an orchestration service without writing a single line of code, but instead by defining how data is transferred between services using a graphical diagram in the Visual Programming Language environment.

A typical screenshot of VPL is shown in Figure 1-3. In VPL this is called a diagram.
Part I: Robotics Developer Studio Fundamentals

Each block represents a service, a calculation, a condition, or a nested diagram, and these are called *activities*. The lines between the blocks represent messages flowing from one service to another. The data flow model works well to represent the messages that are passed in a typical DSS application. VPL diagrams can be arbitrarily complex because each block can represent another nested diagram, which may in turn have other nested diagrams.

*VPL is covered in detail in Chapters 10 through 12.*

**Robotics Samples**

In addition to the system components previously mentioned, MRDS also includes a number of services tailored to work with specific robots, including the LEGO NXT, the iRobot Create and Roomba, the Kondo KHR-1, the MobileRobots Pioneer 3DX, and the Parallax Boe-Bot. Various FischerTechnik robot models are also supported.

The Parallax Boe-Bot, the iRobot Create, and the Pioneer 3DX are shown in Figure 1-4 (not to scale).
Services for various other sensors such as a laser range finder (the large object on top of the Pioneer 3DX in Figure 1-4 on the right), infrared sensors (sticking out at the front of the Boe-Bot on the left), a camera, and a game controller are also included.

**Packages**

In addition to the major 1.0 and 1.5 releases, the MRDS team has also released packages of code targeted to specific scenarios. Each of the three packages is described in the following sections. Instructions for installing the packages are included in the next section, which describes how to install MRDS. All of these packages are available from the MRDS Downloads page at [http://msdn2.microsoft.com/en-us/robotics/aa731520.aspx](http://msdn2.microsoft.com/en-us/robotics/aa731520.aspx).

**The Sumo Package**

The sumo competition is a classic robotics contest. The MRDS team created a sumo competition for the 2007 MEDC conference in Las Vegas, Nevada, using iRobot Create robots.

A sumo competition takes place between two robots inside a circular ring. The objective is for each robot to push its opponent out of the ring without going outside itself. Each robot has sensors on its underside to detect a colored region around the outer edge of the ring. Each also has front-left and front-right bumpers and a camera.

As developers arrived at the conference, they were encouraged to implement a service to control a sumo player. The service was qualified against another basic sumo player service in the simulator and developers that passed the qualification went on to port their services to the actual hardware to compete in a real sumo ring.

It sounds like a simple contest to run but no one anticipated how much work it would be to assemble nearly 50 robots on the day before the conference. Figure 1-5 shows two of the completed sumobots at the end of that very long day.
The custom-built packages on top of the robots contain embedded PCs running Windows CE with MRDS and attached web cameras so that the robots can find each other. The built-in sensors on the Creates are used to detect the edge of the sumo ring.

The sumo package contains a simulated referee service and all of the simulation services needed to run the simulated sumo competition. The sample sumo player service is a good example of an orchestration service that interacts with other MRDS services to read sensor data and control the behavior of the robot. The referee is a different type of orchestration service that starts the matches, runs a timer, and determines when one of the sumobots has won.

The sumo package also includes code that enables the sumo player services to run on the actual hardware. Instructions for building the sumo hardware can be found at http://msdn2.microsoft.com/en-us/robotics/bb403184.aspx.

Chapter 9 discusses how to simulate a sumo competition.

The Soccer Package

The soccer package was developed for RoboCup 2007 in Atlanta. The stated goal of RoboCup is “By the year 2050, develop a team of fully autonomous humanoid robots that can win against the human world soccer champion team.” You can find more information about RoboCup at www.robocup.org.

The soccer package is a simulation-only competition. It includes a soccer referee service as well as a soccer field simulation environment and a sample soccer player service. Each soccer team consists of a field player and a goalkeeper. The LEGO NXT is used for the robots in the MRDS package but the environment is designed to allow other robots. In fact, the robot models used in the RoboCup competition were RobuDog models provided by RoboSoft. These articulated robot dogs were challenging to program but the resulting matches were very entertaining. Figure 1-6 shows the soccer simulation running with the RobuDog models.
Chapter 1: Exploring Microsoft Robotics Developer Studio


Chapter 9 demonstrates how to use the soccer simulation with the simulated Corobot robot developed in Chapter 6.

The Courseware Package

The courseware package contains a set of labs that can be used as part of an introductory course on robotics. They illustrate common robotics problems such as reading sensors and controlling motors. The package includes the following labs:

- Lab Tutorial 1 (VPL): Joystick a Robot in Simulation and in Hardware
- Lab Tutorial 2 (VPL): Advanced Motion
- Lab Tutorial 3 (VPL): Sensing and Simple Behaviors
- Lab Tutorial 4 Part 1 (VPL/C#): Play Mastermind with the Robot via Bi-directional Speech
- Lab Tutorial 4 Part 2 (VPL/C#): Play Mastermind with the Robot via Vision and Text-To-Speech
- Lab Tutorial 5: Using Vision to Estimate the Distance to an Object
- Lab Tutorial 6 (VPL): Task Learning via Human-Robot Interaction
- Lab Tutorial 7 (C#): Multirobot Coordination
In addition to providing several good examples of MRDS services, the courseware package includes a number of services that may be useful in your own robotics projects.

Support for MRDS

Many companies and organizations have done some very interesting things with MRDS. The following sections describe a few examples of projects and products that support MRDS.

The SubjuGator

In July of 2007, students from the University of Florida Machine Intelligence Laboratory entered an autonomous robotic submarine called SubjuGator in the AUVSI/ONR 10th International Autonomous Underwater Vehicle Competition, where they won first place. This was the sixth version of the SubjuGator built by the university and this one had more powerful sensors and a faster computer system than previous versions. In addition, the team used Microsoft Robotics Studio for control and simulation of the submarine.

The SubjuGator was designed to operate underwater at depths of up to 100 feet. A single-board Intel Core 2 Duo-based computer running Windows XP provided the computational horsepower for monitoring and controlling all systems. The mission behavior of the submarine was controlled with Microsoft Robotics Studio communicating with a network of intelligent sensors such as cameras, hydrophones, a Doppler Velocity Log, a digital compass, an altimeter, and internal environment monitor sensors.

The SubjuGator is shown in Figure 1-7. More information about this project can be found at http://subjugator.org.
Prospect 12

Another example of a university project using Microsoft Robotics Studio in a competition is the Princeton Autonomous Vehicle Engineering (PAVE) entry in the 2007 DARPA Urban Challenge. PAVE participated in the Urban Challenge with Prospect 12, a modified Ford Escape Hybrid fitted with several stereo and monocular cameras. PAVE was one of the 35 teams invited to participate in the National Qualification Event, but their vehicle did not compete in the final Urban Challenge race.

According to the PAVE team, their use of Microsoft Robotics Studio had over 25 services running across five dual-core servers. Prospect 12 is shown in Figure 1-8. More information about this project can be found at http://pave.princeton.edu/main/urban-challenge. Specific information about the Prospect 12 can be found at http://pave.princeton.edu/main/urban-challenge/msrs.

![Figure 1-8](image)

The uBot-5

Speaking of university projects, one of the most interesting robotic platforms to run Microsoft Robotics Developer Studio is the uBot-5 developed at the Laboratory for Perceptual Robotics at the University of Massachusetts at Amherst. This robot is a small, lightweight research platform for mobile manipulation. It balances on two wheels using an inverse pendulum model like the Segway Scooter. The balancing is handled by a custom 12-channel FPGA-based servo controller that can update the position and velocity of all the motors at a rate of more than 2 kilohertz. An embedded PowerPC core manages low-level reflexes such as balancing and transitions between postural modes.
The uBot-5 team implemented MRDS services to control the movement of the uBot-5 and they have a simulation model that correctly balances the robot in the simulation environment. The uBot-5 is shown in Figure 1-9. More information is available at [www-robotics.cs.umass.edu/Robots/UBot-5](http://www-robotics.cs.umass.edu/Robots/UBot-5).

![uBot-5](image)

**Figure 1-9**

**KUKA Educational Framework**

KUKA is a company well known for its high-end industrial robots, but they have also been very active in their support for Microsoft Robotics Developer Studio. The first robotic arm implemented in the MRDS Simulator, the KUKA LBR3, was developed with assistance from KUKA.

They have released their own package for MRDS called the KUKA Educational Framework. The framework includes four arm tutorials that teach about point-to-point motions and the orchestration of linked motions using the simulated LBR3. The three mobile tutorials demonstrate how to control a robotic arm attached to a mobile platform. It culminates in a task tutorial that shows how to use the mobile arm equipped with a laser range finder to pick up a box on a table and drop it into a garbage can.

These tutorials can be downloaded at no charge from [www.kuka.com/usa/en/products/software/educational_framework](http://www.kuka.com/usa/en/products/software/educational_framework). A scene from one of the mobile tutorials is shown in Figure 1-10.
Many other companies provide MRDS support for their robots and even general-purpose services. More MRDS partners are listed on the Microsoft Robotics Community page at http://msdn2.microsoft.com/en-us/robotics/aa731519.aspx.

**Setting Up Your System**

This section contains instructions for preparing your system for MRDS and installing the SDK, including all the optional packages. Once you have installed all the software, follow the instructions in the last section to verify that it is working properly.

**System Requirements**

According to the MRDS Data Sheet, application development for MRDS is supported on the following operating systems:

- Windows Vista
- Windows XP
- Windows Server 2003 R2 (32-bit x86)
- Windows XP 64-bit Windows Server 2003 R2 x64 editions

Note that the simulation environment is not supported on 64-bit Windows but all of the other components will work properly.

The run-time and other MRDS services can be deployed and executed on the following operating systems:

- Windows Vista
- Windows XP
Part I: Robotics Developer Studio Fundamentals

- Windows XP Embedded
- Windows Embedded CE 6.0
- Windows Mobile 6.0
- Windows Server 2003 R2 (32-bit x86)
- Windows Server 2003 R2 x64 editions
- Windows XP 64-bit

*Neither Linux nor any other non-Windows operating system is supported at this time.*

The simulation environment has more rigid hardware requirements because it uses 3D graphics extensively. Several developers have had frustrating experiences because they attempted to run the simulator on systems with outdated or underpowered graphics cards.

The “Hardware Requirements” section in Chapter 5 goes into more detail about the graphics card requirements, but the bottom line is that your graphics card must support DirectX 9 graphics using Vertex and Pixel Shader Model 2.0 or higher. Most systems sold since 2006 with reasonable 3D graphics support meet these requirements. There are some lower-end systems with integrated graphics chips, such as the Intel 865G chipset family, that do not meet these requirements. This is one of the most frequently asked questions in the MRDS simulation forum, so it is a good idea to check your graphics card capabilities on the manufacturer’s website if you are having trouble running simulation.

**Prerequisites**

The most important software prerequisite is an appropriate version of Microsoft Visual Studio or Microsoft Visual C# Express. If you use Visual Studio, you should have at least Visual Studio 2005 with Service Pack 1 installed.

If you don’t already have a copy of Visual Studio, you can download Visual C# Express at no charge from [www.microsoft.com/express/vcsharp/Default.aspx](http://www.microsoft.com/express/vcsharp/Default.aspx). This development environment contains all of the components necessary to build MRDS services for all operating systems except Windows CE and Windows Mobile.

Another useful utility to install is the .NET Reflector, which can be downloaded from [www.aisto.com/roeder/dotnet](http://www.aisto.com/roeder/dotnet). Reflector enables you to easily view, navigate, search, decompile, and analyze .NET assemblies.

**Installing MRDS 1.5**

Microsoft Robotics Developer Studio version 1.5 (December 2007 Refresh) is available as a web download from [www.microsoft.com/robotics](http://www.microsoft.com/robotics) at the time of writing. It is highly likely that a new version will be available as this book goes to press, but you should stick with V1.5 Refresh, which is an official release, not a beta.

As noted in the introduction, this book was written using version 1.5. Most of the services will probably work with MRDS V2.0 when it is released, provided that you first convert them using the DssProjectMigration tool. However, the authors cannot make any guarantees about this, so you must install MRDS V1.5 to ensure that the examples work. Updates on migration issues will be posted to the book’s website.
Chapter 1: Exploring Microsoft Robotics Developer Studio

This is a critical point because services written to work with version 1.5 of the SDK will not work with later versions. This is not because the MRDS team has broken backward compatibility but because MRDS services use strong-name signing, and when a service has a reference to another service it will not run if the second service has a different version than when the first service was built. This can usually be fixed by rebuilding any services that have references to run-time services that have changed in the new release, but obviously this isn’t possible if you don’t have the source code.

This means that the sample code for this book will not automatically work with later releases of MRDS. The authors intend to provide updated sample code for the book as Microsoft releases new versions of MRDS. You can find updated sample code at www.wrox.com or at the authors’ website at www.proMRDS.com. Keep watching the website for revised versions of the examples soon after MRDS V2.0 ships.

**Strong-Name Signing Difficulties**

The problem of signed assemblies has also caused some difficulties for developers as they are exploring MRDS for the first time. Consider Simulation Tutorial 2, for example. It uses the SimpleDashboard service to drive the robots around in the simulation environment. The SimpleDashboard service has a reference to the SickLRF service. If you happen to rebuild the SickLRF service, which is also provided as a sample, the SimpleDashboard service won’t run because the SickLRF service has now been signed with a different key than when it was originally built by Microsoft and the SimpleDashboard service won’t load it due to security concerns. This can be quickly fixed by rebuilding the SimpleDashboard service, but then any other services that use the SimpleDashboard service must also be rebuilt. Fortunately, there are no other Microsoft services that reference the SimpleDashboard service because it is a top-level orchestration service.

The robotics team has tried to minimize this problem by building solutions for each of the tutorials, including all of the services that the tutorial depends on. Simply rebuilding the entire solution generally fixes the problem. One bug in the 1.5 release is that the SimpleDashboard service was not included in the solution for Simulation Tutorial 2.

In December 2007, the MRDS team released a refresh of the 1.5 release with bug fixes and some minor new functionality. If you downloaded version 1.5 of the SDK prior to December 2007, you should uninstall MRDS and then download and reinstall the refreshed kit. When you uninstall MRDS, it only removes the files that it installed. If you have added any of your own services to the Microsoft Robotics Studio directory structure, they will not be deleted.

If you have version 1.0 or any of the Community Technology Preview releases of 1.5 installed, it is not necessary to uninstall them. They can coexist alongside the 1.5 release. However, you should probably remove them because they are obsolete.

Using your web browser, navigate to www.microsoft.com/robotics and select the Downloads link from the right side of the page. All of the currently available downloads for MRDS are listed on this page. Click Microsoft Robotics Studio (1.5) Refresh and click Download on the next page that appears. The size of this download is 87.5MB.
It is recommended that you save the file to your hard disk and then run it after it has been downloaded. When you run the executable, the MRDS installer will run as shown in Figure 1-11.

After you accept the licensing agreement (notice that MRDS is free for noncommercial use but must be licensed for commercial use), the MRDS files are installed. After the InstallShield Wizard has completed, the installer gives you options for installing additional components. Be sure to check each of these options and install them as shown in Figure 1-12.
DirectX, XNA, and the AGEIA PhysX engine are required to run the simulation environment, and .NET 3.0 is required to run VPL. It is also nice to have the MRDS help files integrated into Visual Studio so that help for these APIs is visible there. Be sure that you have an active Internet connection when you run these installers because the DirectX and XNA installers get the latest installation kits from the Web.

**Robotics Studio 1.5 Directory Structure**

By default, the MRDS 1.5 SDK is installed to C:\Microsoft Robotics Studio (1.5). Some of the more important directories under this main directory are described in this section.

All the user and run-time service assemblies are stored in the bin directory. This directory should be in your DOS path when you open an MRDS command window. Assemblies that target the .NET Compact Framework (CF) are in the CF subdirectory.

As you might expect, the documentation directory contains all of the SDK documentation. The external directory contains installer programs for other software that MRDS depends on. This software should be installed when you install MRDS, but in case there was a problem, you can find the installers here.

The store directory contains content and data files used by MRDS. The contract directory cache is stored in this directory. The store\media directory contains all of the textures and meshes used by the simulator.

Most of the remaining files in the SDK are in the samples directory. The major subdirectories are as follows:

- **Common**: This directory contains the source files for the RoboticsCommon assembly, which contains generic contracts for many types of devices.
- **Config**: This directory contains manifests and configuration files for all of the samples.
- **Diagrams**: This directory contains sample VPL diagrams.
- **Misc**: This directory contains source code for services such as the SimpleDashboard service, the GameController service, and many others.
- **Platforms**: This directory contains subdirectories for each of the supported platforms: FischerTechnik, iRobot, Kondo, LEGO, MobileRobots, and Parallax.
- **Sensors**: This directory contains subdirectories for sensor services, among which are the SickLRF and the Webcam services.
- **Simulation**: This directory contains subdirectories for all of the simulation-specific services, such as the SimulatedDifferentialDrive, simulated sensors, the source code for all of the simulation entities, and the ArticulatedArms services.
- **Technologies**: This directory contains subdirectories that have source code for services that encapsulate specific technologies such as Speech.
- **Test**: This directory contains all of the test sample code demonstrating how to write service tests for your services.
- **HostingTutorials, RoboticsTutorials, ServiceTutorials, SimulationTutorials, and VPLTutorials**: These directories contain the source code for all of the tutorials described in the MRDS documentation.
An unusual feature of the SDK is that all services and files must be contained in this directory structure. It is recommended that you create a directory under the root directory with your company or organization name to hold your projects. Some developers have made the mistake of creating their own projects under the samples directory because that is where the source code for all of the other services is kept.

**Installing the Packages**

As long as you are at the MRDS Downloads page, you may as well download and install the sumo package, the soccer package, and the courseware package for MRDS 1.5.

Packages do not contain a full installer. They are generated by a DSS utility program call DssDeploy. This utility gathers all of the files needed to run a particular *manifest* (a list of all the required services and their partnerships), along with any additional files specified, compresses them, and packs them into an executable. When the package is executed, it may display an EULA (end-user license agreement) that must be accepted, after which it installs the files. Finally, it displays an optional ReadMe web page that provides further information about the package.

It is recommended that you install all three packages. Some of the examples in this book assume that this code is installed. Figure 1-13 shows the sumo package being installed.

Packages are usually targeted for a specific release of MRDS. A package that is targeted to the 1.5 SDK will fail to install if that version of the SDK is not also installed.

When you have installed a package, an HTML file is placed in the *packages* directory, which contains a list of all the files that were installed as part of the package.
Chapter 1: Exploring Microsoft Robotics Developer Studio

Installing the Sample Code for This Book

Unless you enjoy typing in hundreds of lines of code, download the sample code package associated with this book. All of the source code used in this book, along with other necessary media files, is available for download at www.wrox.com. At the site, simply locate the book’s title (either by using the Search box or by using one of the title lists) and click the Download Code link on the book’s detail page to obtain all the source code for the book.

The downloaded file is a DssDeploy package just like the MRDS packages. Install it the same way that you installed the other packages. The ReadMe page that is displayed at the end of the installation provides more information about the source code, along with links to pages for each chapter.

The source code is installed in a subdirectory called ProMRDS that is placed in the MRDS top-level directory. Each chapter has its own subdirectory with more subdirectories for each project within the chapter. Binaries for the samples services, along with command files to run the samples, are installed in the bin directory. Media files required by the simulation samples are installed in the store\media directory.

Verifying the Installation

Here are a couple of quick checks you can use to verify that you have installed everything correctly and the software is working properly on your machine.

Runtime

Open a MRDS Command Prompt window by clicking Start ➔ All Programs ➔ Microsoft Robotics Studio (1.5) ➔ Command Prompt. This brings up a DOS window that has the Path environment variable set up properly for running applications. Type the following command, as shown in Figure 1-14, and you should see a similar output: dsshost -p:50000 -t:50001.
This starts up a DSS node. Verify that the node is running properly by pointing a web browser to http://localhost:50000. Click Service Directory in the left panel and verify that the services running are the same as those shown in Figure 1-15.

![Service Instance Directory](image)

**Figure 1-15**

This verifies that the DSS runtime is properly installed. You can terminate the DSS node by pressing Ctrl+C in the Command Prompt window.

**Simulation**

To verify that the simulation environment is working properly on your machine, first make sure that you have stopped the DSS node that was started in the previous section. This test runs a DSS node using the same ports, so it won’t run if the previous node is still running. Click Start ➔ All Programs ➔ Microsoft Robotics Studio (1.5) ➔ Visual Simulation Environment ➔ Basic Simulation Environment.

After a few moments, a window similar to the one shown in Figure 1-16 should appear. Click View ➔ Select Status Bar to display the frame rate. The frame rate should be somewhere near the refresh rate for your display. If it is less than 10 frames per second (FPS), you need a more powerful graphics card.
If your window is red and contains an error message, shows a solid light-blue color, or looks substantially different from the scene shown in Figure 1-16, check the capabilities of your graphics card.

**ProMRDS Sample Code**

A very simple little sample program is provided to verify that you have properly installed the sample code associated with this book. Make sure that all DSS nodes have been terminated and then open an MRDS Command Prompt window and type **Welcome**, as shown in Figure 1-17.
You should see the output shown in Figure 1-17, and after a moment or two a dialog box similar to the one shown in Figure 1-18 should appear. If your speaker volume is on and speech services are correctly configured on your machine, you should hear the words in the dialog box spoken by the computer.

![Figure 1-18](image)

This verifies that the source code for the book was properly installed. Incidentally, that very simple service was generated with the VPL diagram shown in Figure 1-19. More great VPL examples can be found in Chapters 10 through 12. You can terminate the DSS node by pressing Ctrl+C in the Command Prompt window.

![Figure 1-19](image)

For an example that shows off the power of MRDS, try a different service. Follow these steps:

1. In Windows Explorer, navigate to the MRDS installation point and then to ProMRDS\Chapter9\ExplorerSim.
2. Open ExplorerSim.sln in Visual Studio by double-clicking it.

3. Once it is open, run the service with the debugger by pressing F5 or clicking Debug ➤ Start Debugging. After a little while, a DOS Command Prompt window appears, followed by a simulation window, shown in Figure 1-20.

![Figure 1-20](image)

The simulated Pioneer 3DX robot wanders around building a map using information from the simulated laser range finder. You can see the map in the Map window. Figure 1-21 shows a map after the robot has done a fair amount of exploring.

![Figure 1-21](image)

A Dashboard window also appears when you run the service.
Part I: Robotics Developer Studio Fundamentals

4. Enter **localhost** as the Machine and **50001** as the Port and click Connect. Three services are displayed in the service directory list. Double-click simulatedlrf to select it. Figure 1-22 shows the Dashboard with the Laser Range Finder information displayed.

![Figure 1-22](image)

5. Double-click the simulatedwebcam service. A webcam window pops up to show the robot’s view of the world (see Figure 1-23).

![Figure 1-23](image)

The ExplorerSim service illustrates many different aspects of MRDS and it is interesting to watch for a while. When you get tired of it, just stop the program in Visual Studio.
Additional Resources and Support

It should be clear at this point that there is a great deal to learn about Microsoft Robotics Developer Studio. This book gives you a great start, but it is best to take advantage of all the resources available. The following sections point out a few of these.

www.Microsoft.com/Robotics

This is Microsoft Robotics Developer Studio central. You can find information about the latest releases on this page along with pointers to many other articles and blogs written by other users. The Community link has links to partner websites where MRDS-related products are offered. There are also links to MRDS-related user groups and discussion boards.

On the main MRDS page, you can also find a link to the MRDS wiki, which contains hints, bug workarounds, and additional information about using MRDS.

Add this page to your Favorites list, as you will likely refer back to it often.

Tutorials and User Guides

Following a long Microsoft tradition, the early documentation provided with Microsoft Robotics Developer Studio was a little thin. It is rapidly becoming more complete and now includes several informative user guides along with the tutorials.

The MRDS documentation can be accessed from a link on the Start menu. The Samples and Tutorials Overview at the beginning of the documentation provides a good roadmap for exploring the tutorials. It is strongly recommended that you do all of the tutorials. Start with the DSS tutorials to learn service programming by example. These tutorials are good companion material to Chapters 2–4 in this book. Likewise, the Simulation tutorials provide a different view of the material in Chapters 5–9. The VPL tutorials go with the material in Chapters 10–12, and the Robotics tutorials supplement the information in Chapters 13–17.

In addition to the samples and tutorials, CCR and DSS each have a good User Guide that is available in the Microsoft Robotics Developer Studio Runtime section in the MRDS documentation.

Additional sections of interest include the Courseware section and the Technology Samples section.

Online Forums

Sometimes you might run into a problem that just can’t be answered by this book or the tutorials and samples. In this case, you should turn to the online forums. There is a link on the main MRDS page to get to the forums, or you can go directly to http://forums.microsoft.com/msdn/default.aspx?forumgroupid=383&siteid=1.

The forums provide an opportunity to post questions in the following six categories:

- Community: Samples, feedback, and general discussion
- Concurrency and Coordination Runtime: Everything about the CCR
Part I: Robotics Developer Studio Fundamentals

- Decentralized Software Services: Everything about DSS
- Simulation: Samples, entities, and graphics/physics discussion
- Visual Programming Language: Everything about VPL
- Hardware Configuration and Troubleshooting: How to set up your hardware and help with porting code to MRDS

Members of the MRDS team spend considerable time answering questions on the forums, and other users often contribute answers. It is worthwhile to search through past threads before posting a new question on the forums.

*If you are posting code to the forum, try to condense it down to the shortest piece of code that illustrates the problem. Copy and paste it into the posting window, and then select it and mark it as code by clicking the Mark Code Block button in the editing toolbar. This prevents your code from appearing in the forum with a bunch of smiley faces in it.*

**Channel 9**

Channel 9 is a Microsoft-sponsored developer community with many interesting links to other developers’ MRDS projects. The Channel 9 home page is [http://channel9.msdn.com](http://channel9.msdn.com) and from there you can click the Search link to search for videos, projects, and posts about MRDS. There are also links to several classic Channel 9 videos, available directly from the MRDS home page in the Featured Videos section.

Take a look at some of the projects presented and be sure to post your own project in the Sandbox when it is ready for the world to see.

**Summary**

So that’s Microsoft Robotics Developer Studio in a nutshell. Now that you’ve seen the opportunities that await you, it’s time to warm up your compiler and get coding. First one to write a sentient service wins!