

# Developing GIS Solutions with MapObjects™ and Visual Basic®

---



**Bruce A. Ralston**

*Learn to Program GIS Applications from the Ground Up*



# Developing GIS Solutions with MapObjects™ and Visual Basic®

*Bruce A. Ralston*

**ONWORD PRESS**  
—★—  
**THOMSON LEARNING**

---

Australia Canada Mexico Singapore United Kingdom United States

# Developing GIS Solutions with MapObjects™ and Visual Basic®

By Bruce A. Ralston

**Publisher:**  
Alar Elken

**Executive Editor:**  
Sandy Clark

**Acquisitions Editor:**  
James Gish

**Managing Editor:**  
Carol Leyba

**Development Editor:**  
Daril Bentley

**Editorial Assistant:**  
Jaimie Wetzel

**Executive Marketing Manager:**  
Maura Theriault

**Executive Production Manager:**  
Mary Ellen Black

**Production Manager:**  
Larry Main

**Manufacturing Coordinator:**  
Betsy Hough

**Technology Project Manager:**  
David Porush

**Cover Design:**  
Cammi Noah

## Trademarks

MapObjects is a trademark of Environmental Systems Research Institute (ESRI), Inc. Visual Basic is a registered trademark of Microsoft, Inc. ArcView GIS and ArcInfo are registered trademarks of ESRI, Inc.

Copyright © 2002 by OnWord Press.  
OnWord Press is an imprint of  
Thomson Learning  
SAN 694-0269

Printed in Canada  
10 9 8 7 6 5 4 3 2 1

For permission to use material from  
this text, contact us by  
Tel : 1-800-730-2214  
Fax: 1-800-730-2215  
[www.thomsonrights.com](http://www.thomsonrights.com)

For more information, contact:  
OnWord Press  
An imprint of Thomson Learning  
Box 15-015  
Albany, New York 12212-15015

Or find us on the World Wide Web at  
<http://www.onwordpress.com>

Library of Congress  
Cataloging-in-Publication Data  
is available for this title  
ISBN: 0-7668-5438-8

**All rights reserved.** No part of this work covered by the copyright hereon may be reproduced or used in any form or by any means—graphic, electronic or mechanical, including photocopying, recording, taping, Web distribution or information storage and retrieval systems—without the written permission of the publisher.

## NOTICE TO THE READER

Publisher does not warrant or guarantee any of the products described herein or perform any independent analysis in connection with any of the product information contained herein. Publisher does not assume, and expressly disclaims, any obligation to obtain and include information other than that provided to it by the manufacturer.

The reader is expressly warned to consider and adopt all safety precautions that might be indicated by the activities herein and to avoid all potential hazards. By following the instructions contained herein, the reader willingly assumes all risks in connection with such instructions.

The publisher makes no representation or warranties of any kind, including but not limited to, the warranties of fitness for particular purpose or merchantability, nor are any such representations implied with respect to the material set forth herein, and the publisher takes no responsibility with respect to such material. The publisher shall not be liable for any special, consequential, or exemplary damages resulting, in whole or part, from the readers' use of, or reliance upon, this material.

---

## ■ ■ About the Author

Dr. Bruce Ralston is a professor in and head of the Department of Geography at the University of Tennessee, Knoxville. He holds multiple awards, including the Edward L. Ullman Award for Outstanding Contributions to the Field of Transportation Geography, and the Applied Geography Award, both from the Association of American Geographers. He is also a consultant to various public and private sector organizations and President of GIS Tools, Inc.

## ■ ■ Acknowledgments

I want to thank the many students who had the desire to learn this material. One of the luckiest things that can happen to a professor is to have students whose enthusiasm for learning pushes him to also learn. I have been extremely fortunate at the University of Tennessee to have a steady stream of such students. They have helped me learn many new areas, including the topics covered in this book. Grateful acknowledgment is also made to the team at OnWord Press.





# Introduction

If you have ever tried to learn a foreign language, you know how helpful a dictionary can be. You also know that it is virtually impossible to master a language simply by studying a dictionary or taking classes. Ultimately, you have to immerse yourself in the culture, finding yourself in situations in which you have to use the language to be successful, such as ordering a meal or obtaining directions. A dictionary can help, but it is not enough.

## ■ ■ Purpose and Approach

The aforementioned is the problem with help files and, quite frankly, a lot of programming books. They are more like dictionaries. They help you look up specific things once you are conversant in the language, but they do not put you into the culture. Even the example programs in the MapObjects Help file, although useful, provide only “snippets of conversations,” not a complete discourse in the original language.

This book attempts to “get you into” the Visual Basic/MapObjects culture. The book begins by examining some simple programs, and by Chapter 4 you will be building a program—a conversation with a user—that becomes more complex in subsequent chapters. In most chapters you will learn a new aspect of Visual Basic programming and then use that knowledge to exploit the potential of MapObjects.

## ■ ■ Audience and Prerequisites

This book was developed for users of GIS who want to learn how to develop their own applications using Visual Basic and MapObjects. Although basic aspects of Visual Basic programming are pre-

sented in the text, this is not meant to be a comprehensive introduction to computer programming. The goal of the book is to help readers learn enough about VB and MO so that they can develop their own programs and implement them as either stand-alone applications or as the basis for web-based GIS. Experience with other ESRI products, such as Arc or ArcView, is helpful.

There are some software prerequisites for using this book. Readers should have access to Visual Basic 6, MapObjects 2.0 or later, and (if you wish to implement the web-based application described in Chapter 15) MapObjects IMS. Readers who do not have these ESRI products can download 90-day evaluation copies at [www.esri.com/software/mapobjects/download.html](http://www.esri.com/software/mapobjects/download.html) and [www.esri.com/software/mapobjects/ims/eval.html](http://www.esri.com/software/mapobjects/ims/eval.html). Chapter 14 requires the user to have a web server application that supports PERL scripts.

## ■ ■ How To Use This Book

The book contains a series of example programs that build from a very simple map display application in Chapter 4 to advanced applications in later chapters. Starting with Chapter 4, each succeeding chapter will add new functionality to the program developed in the previous chapters. Each program developed in the text can be found on the companion CD-ROM, so that readers can either enter the code that is described in the text or load the corresponding sample programs from the companion CD-ROM.

## ■ ■ Content and Structure

This book can be broken into four parts. Chapters 1 through 3 provide a brief introduction to programming with Active X components and Visual Basic (VB). Readers familiar with VB may wish to skim these chapters. The remaining chapters deal with developing GIS programs. In each chapter, a new aspect of VB is discussed, along with new GIS properties that can be found in MapObjects (MO). The VB discussed in the chapter is then combined with MO objects to add more GIS functionality to the program developed in the chapters. Chapters 4 through 7 deal with managing map layers.

Chapter 7 deals with identifying elements of map layers and working with selected sets. Chapters 8 through 11 cover thematic mapping. Chapter 12 deals with collections, classes, and theme-on-theme selections. Chapters 13 through 15 cover web-based GIS. Chapter 13 provides a brief introduction to HTML, whereas Chapters 14 and 15 present two approaches to serving maps on the Web. The final three chapters (16 through 18) deal with map projections, coordinate systems, and buffering and overlay.

## Book Features and Conventions

This edition includes a companion CD-ROM at the back of the book (see “About the Companion CD-ROM” at the end of this introduction). Throughout the book you will see references to the companion CD-ROM. It is there that you will find VB projects, data sets, DLLs, and other utilities discussed in the text.

Italic font in regular text is used to distinguish certain command names, code elements, file names, directory and path names, user input, and similar items. Italic is also used to highlight terms and for emphasis.

The following is an example of the monospaced font used for code examples (i.e., command statements) and computer/operating system responses, as well as passages of programming script.

```
var myimage = InternetExplorer ? parent.  
cell : parent.document.embeds[0];
```

The following are the design conventions used for various “working parts” of the text. In addition to these, you will find that the text incorporates many exercises and examples.



**NOTE:** Information on features and tasks that requires emphasis or that is not immediately obvious appears in notes.



**TIP:** Tips on command usage, shortcuts, and other information aimed at saving you time and work appear like this.



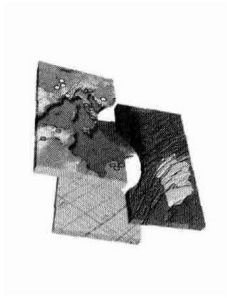
**CD-ROM NOTE:** These notes point to files and directories on the companion CD-ROM that supplement the text via visual examples and further information on a particular topic.

## ■ ■ About the Companion CD-ROM

The companion CD-ROM consists of four directories, each of which contains material used in the text. Each directory and its content are described in the following.

- ❑ *MO2*: This directory contains all VB projects referenced in the text. Whenever you see a CD-ROM Note that references a particular directory, it will be found under this directory. These projects can be opened and executed in Visual Basic.
- ❑ *Utility*: This directory contains a DLL for converting bit-map files to *jpg* files. It is used in Chapter 14 to develop web-based GIS without a commercial IMS product.
- ❑ *Web*: This directory contains materials used for web-based GIS. These include HTML pages, two PERL scripts, Active Server Pages (ASP) scripts, and a *jpg* file of Tennessee. The use of these files is discussed in Chapters 13 through 15.
- ❑ *Shapes*: The shapes used in the examples are stored here. There are two directories under the *Shapes* directory. *USA* contains shapes for the United States, and *Knox* contains shapes for Knox County, Tennessee. The shapes in the *USA* subdirectory are used in Chapters 4 through 15, and the shapes in the *Knox* subdirectory are used in chapters 16 through 18.





# Contents

<b>Chapter 1: Entering the VB/MO Culture</b>	<b>1</b>
Introduction	1
An Introduction to OOP Implementations	1
Working in Visual Basic	5
Forms and Controls	13
The VB Controls Sample Program	16
Check Boxes, Radio Buttons, and List and Combo Boxes	19
Summary	24
<b>Chapter 2: Using ActiveX Controls</b>	<b>25</b>
Introduction	25
A Data Access Application	25
Adding DLL Controls	29
A Closer Look at VB Projects	32
Summary	32
<b>Chapter 3: Programming Basics</b>	<b>33</b>
Introduction	33
Types of Code Modules in VB	33
The Mechanics of Editing Code	35
Variables	37
Variable Scope	39
Data Types	40
Procedures	45
Control Structures and Message Boxes	48
Summary	54
<b>Chapter 4: The MapObjects ActiveX Control</b>	<b>55</b>
Introduction	55
ActiveX Components and DLLs	55
The MapObjects ActiveX Control	57
Working with Layers	63
Summary	66

---

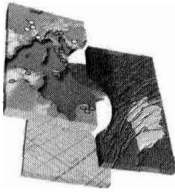
<b>Chapter 5: Managing Map Layers</b>	<b>.67</b>
Introduction	67
VB's Common Dialog	68
Data Connections, GeoData Sets, and Adding Layers and Images	69
Using the CommonDialog Object with GeoData Sets	72
MoView2: A Helpful Reference	73
Adding Layers Interactively	73
Synchronizing the Map and the Add Layer Form	80
Summary	80
<b>Chapter 6: Toolbars and Layer Management</b>	<b>.81</b>
Introduction	81
Adding a Remove Button	81
Image Lists and Toolbars	84
The Map Toolbar	89
Zoom Methods	91
Summary	92
<b>Chapter 7: Geometry, Coordinates, and Identifying Features</b>	<b>.93</b>
Introduction	93
Distance Methods	93
Resizing the Main Form	95
The MapObjects Recordset Object	96
Managing the Identify Button	99
Retrieving a Selection	103
Displaying Selected Records	106
Summary	110
<b>Chapter 8: Rendering, Part 1: Single Symbols</b>	<b>.111</b>
Introduction	111
The Tab Strip Control	111
Setting Drawing Properties	115
Summary	120
<b>Chapter 9: Rendering, Part 2: The Unique-value Map Renderer</b>	<b>.121</b>
Introduction	121
VB and Classes	123
Editing the frmDrawProps Code Page	125
Creating the Value Map Renderer	128
Summary	136
<b>Chapter 10: The Unique-value Map Renderer Continued</b>	<b>.137</b>
Introduction	137
Changing the Symbol in a Flex Grid Cell	138
Initializing frmDrawProps	143

---

Displaying the Current Renderer Page .....	144
Summary.....	147
<b>Chapter 11: The Quantile Renderer .....</b>	<b>149</b>
Introduction.....	149
Building the Quantile Renderer.....	149
Restoring the Quantile Renderer.....	158
Summary.....	160
<b>Chapter 12: Collections, Classes, and Advanced Selections .....</b>	<b>161</b>
Introduction.....	161
The Collection Object and the Selection Button .....	162
Enabling Selections .....	163
The Need for a Class .....	167
Using the Class.....	168
Saving the Selected Set .....	172
Selecting by Theme .....	180
Summary.....	186
<b>Chapter 13: Web Basics .....</b>	<b>187</b>
Introduction.....	187
GIS and the Web .....	187
HTML Basics .....	191
Form Basics.....	199
Web Page Data Input Devices .....	200
Summary .....	208
<b>Chapter 14: Serving Maps on the Web: Method 1 .....</b>	<b>209</b>
Introduction.....	209
Modifying Form Units: From Twips to Pixels.....	211
Getting Input File Values .....	212
Translation Functions .....	216
MakeBatchMap: Part 1 .....	218
MakeBatchMap: Part2 .....	222
MakeBatchMap: Part 3 .....	227
Writing the Web Page .....	228
Putting It All Together .....	233
Summary.....	237
<b>Chapter 15: Serving Maps on the Web: Method 2 .....</b>	<b>239</b>
Introduction.....	239
An Overview of the MOIMS .....	240
Working with the WebLink Control.....	241
Configuring the IMS .....	251
A Setup Form .....	254
Server Maintenance.....	255

Comments on the MOIMS .....	255
Summary.....	256
<b>Chapter 16: Buffering and Overlay: Part 1.....</b>	<b>257</b>
Introduction.....	257
Setting Up the Project.....	258
Point Buffers and Coordinate Systems.....	260
The BufferPoints Sub .....	263
Drawing the Buffers.....	265
Specifying Overlay Parameters .....	267
The Intersect Function .....	272
Summary.....	274
<b>Chapter 17: On-the-Fly Projections.....</b>	<b>277</b>
Introduction.....	277
Datums and Geographic Coordinates .....	277
Projection and Geographic Coordinate Systems.....	283
Using Coordinate Systems.....	284
Summary.....	288
<b>Chapter 18: Buffering and Overlay: Part2 .....</b>	<b>289</b>
Introduction.....	289
Allowing Buffers of Any Shape Type .....	290
Buffering .....	293
Reporting the Results .....	296
Tying Up Some Loose Ends .....	301
Summary.....	308

# Chapter 1



# Entering the VB/MO Culture

## ■ ■ Introduction

Developing GIS solutions requires writing code that can manage and manipulate geographic data. Advances in software development tools have accompanied the rapid acceptance of GIS. Two advances in software development have been the development of object-oriented programming (OOP) languages and the use of components for the distribution of objects that can be used with OOP languages. Visual Basic (VB) is a language that allows users to develop and manipulate objects. MapObjects contains a set of objects of interest to GIS application developers. Together, these tools make it possible to develop OOP-based GIS implementations.

## ■ ■ An Introduction to OOP Implementations

To understand OOP you must understand objects. Objects have *properties* and *methods*. It is important that you understand the difference between these. First, however, you need to understand the concepts “class” and “object.” A *class* is an abstract (meaning non-discrete) term, such as *human being*. An *object* is a discrete realization or instance of a class (i.e., by analogy, a particular human being). It is in the class that *properties* and *methods* are defined. Every object (and therefore its properties and methods) is the realization of a class, just as every person is the realization of the concept “human being.”

Toward understanding the difference between properties and methods, let's use another real-world analogy: a bicycle. A bicycle (an "object") has many properties, among them color, size, weight, number of tires, brand, and type of spokes. Applying this analogy to scripting, if you wanted to determine a property (such as color or brand) of the object (bicycle), you would typically use a *Get* command, as in

```
color = bicycle.GetColor()
```

or

```
brand = bicycle.GetBrand()
```

The first instance returns a color, which might be represented by a string (such as BLUE) or a number (such as 0xFF000, which is the hex number—a number in base 16—for orange). The second instance is more likely to return information represented as a string, such as TREK (a brand of bicycle). The point is: *You must know what type of variable your Get request is returning.* At this point you might want to try adjunct exercise 1-1, which follows.

### Adjunct Exercise 1-1: Determining Variables Returned by the Get Request



**NOTE:** This exercise assumes you have ArcView 3.

To determine variables returned by the *Get* request, perform the following steps.

- 1 Start ArcView.
- 2 Go to the help file, click on the INDEX tab, and type *GET*.
- 3 Scroll down and note that there are a lot of entries. Click on the FIND tab, and type *get* (lowercase, as this is case sensitive).

How many "get" topics have been found? Not all of these will be for *Get* requests, but, as you saw from the INDEX tab, a lot of them will be.

You now know that you can find an object's properties (like a map's extent or projection, or a layer's default symbol) with a *Get* request. Curiously, however, not all *Get* requests use the word *Get*!



Returning to our bicycle example, suppose you were ordering a new bike. In this case, you might want to order a specific color. The sister command of *Get* is *Set*, as in

```
bicycle.SetColor(BLUE)
```

Here, the color of the bicycle has been set to *BLUE*. Note that there is no equals sign (or more accurately “assignment operator”) in this statement. (At this point you might want to try adjunct exercise 1-2, which follows.)

### Adjunct Exercise 1-2: Determining Variables Returned by the Set Request



**NOTE:** *This exercise assumes you have ArcView 3.*

To determine variables returned by the *Set* request, perform the following steps.

- 1 Start ArcView.
- 2 Go to the help file, click on the INDEX tab, and type *SET*.
- 3 Scroll down and note that there are a lot of entries. Click on the FIND tab, and type *set* (lowercase, as this is case sensitive).

How many “set” topics have been found? Not all of these will be for *Set* requests, but, as you saw from the INDEX tab, a lot of them will be.

One last thing about *Set*: so far you have been looking at OOP languages in the abstract. Later in the book, you will see in regard to VB that there are two related ways of setting values: *Set* and *Let*. You do not need to worry about the difference between these just yet.

Some properties cannot be changed. For example, in regard to the bicycle example, you might not allow the following:

```
bicycle.SetNumTires(3)
```

Bicycles, by definition, must have two (and not more or less than two) tires. If there are three tires, it is a tricycle, not a bicycle! (One tire, and it is a unicycle.) This points out a very important rule, expressed in the following Note.



**NOTE:** *With some properties you have read (Get) and write (Set) access. With others, you only have read (Get) access. Still others only have write (Set) access.*

Finally, consider the tires themselves. They, too, are objects. They, too, have properties and methods. When you want to inflate the tires, the object of interest is the tire, not the bicycle. That is,

```
bicycle.SetPressure(90)
```

may be illegal, but

```
tires.SetPressure(90)
```

may be fine.

This points out yet another important lesson, expressed in the following Note.



**NOTE:** *You must know which properties and methods go with which object.*

This is often confusing and seemingly counterintuitive, but it is extremely important. Consider another example. Suppose you had an object named *HOUSE*. The house object contains several other objects, one of which might be *BATHTUB*. A command such as

```
bathtub.SetWaterLevel(top)
```

might be fine, but

```
house.SetWaterLevel(top)
```

might be disastrous! However,

```
bathtub.SetCleanToday(TRUE)
```

and

```
house.SetCleanToday(TRUE)
```

might both be legal and desirable.

In summary, you know that there are things called objects, which have properties and that can consist of other objects. You can determine the value of those properties with *Get* commands, and may be able to change those properties with *Set* commands.

Let's turn to methods. Methods are the things objects do. For example, a "person" object might have "eat," "sleep," "move," and "rest" methods. Our "bicycle" object might have "move" and

“stop” methods. A map object can be drawn or scaled. A layer object can be turned off or on. In fact, you have already encountered methods. For example, to redraw a map in VB/MO, you would issue the command *themap.Refresh*.

## ■ Working in Visual Basic

Let's take a look at VB. As stated, VB uses an integrated development environment (IDE) that facilitates a “trial-and-error” approach to program development. That is, VB programs are developed in the IDE and can be tested without compiling the program or leaving the IDE. To start the IDE, you need only start VB (see figure 1-1).



**CD-ROM NOTE:** The VB project in the Chapter 1\_1 directory on the companion CD-ROM starts here.

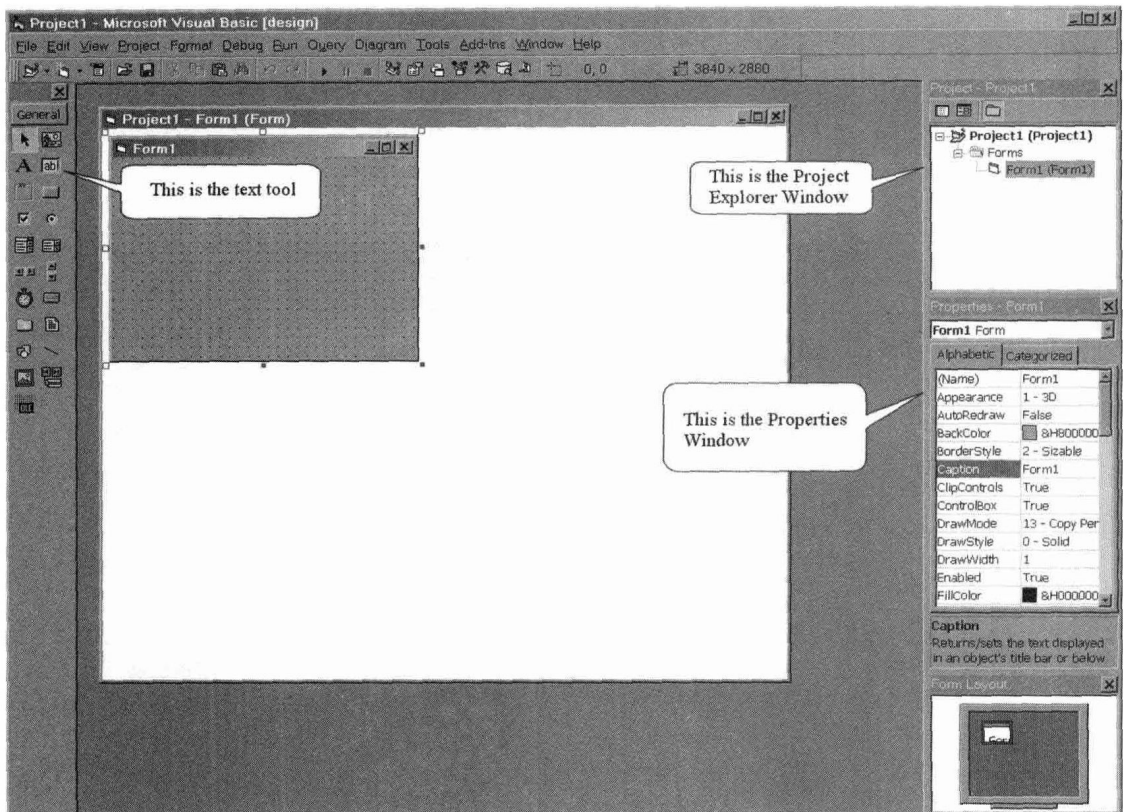


Fig. 1-1. The Integrated Development Environment (IDE).