AutoCAD 2006 VBA

A Programmer's Reference

Joe Sutphin

Apress®

AutoCAD 2006 VBA: A Programmer's Reference

ISBN: 1-59059-579-3

Library of Congress Cataloging-in-Publication data is available upon request.

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

Lead Editor: Tony Davis
Technical Reviewers: Phillip Ash, Steve Johnson
Editorial Board: Steve Anglin, Dan Appleman, Ewan Buckingham, Gary Cornell, Tony Davis, Jason Gilmore, Jonathan Hassell, Chris Mills, Dominic Shakeshaft, Jim Sumser
Associate Publisher: Grace Wong
Project Manager: Kylie Johnston
Copy Edit Manager: Nicole LeClerc
Copy Editors: Candace English, Kim Wimpsett
Assistant Production Director: Kari Brooks-Copony
Production Editor: Janet Vail
Compositor: Linda Weidemann, Wolf Creek Press
Proofreaders: Linda Seifert and Sue Boshers
Indexer: Broccoli Information Management
Interior Designer: Van Winkle Design Group
Cover Designer: Kurt Krames
Manufacturing Manager: 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 2560 Ninth Street, Suite 219, Berkeley, CA 94710. 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 section.

This book is dedicated to my wife, Grace.
Without her, I would not be able to
accomplish the task of writing a book.

Contents at a Glance

Contents

About the Author

JOE SUTPHIN's background includes more than 25 years in the machinery manufacturing industry. He has more than 18 years of CAD experience with 12+ years of AutoCAD-specific experience. Joe is an Autodesk-registered developer, and his work has appeared in the pages of *Cadence* and *Cadalyst* magazines. He has been programming for nearly 20 years, with the last 15 years being Visual Basic–specific experience. In 1998 he collaborated with Microsoft on a Visual Basic application case study. He is the author of the best-selling book *AutoCAD 2000 VBA Programmer's Reference*.

About the Technical Reviewers

PHILLIP ASH began programming in the early 80s when public schools had either Apple IIs or TRS-80s. DOS was loaded from a cassette player, and BASIC was the *lingua franca* of the tyro programming set. He got involved in computer-aided design as a hobby in the late 80s, and after a few semesters in an architectural design and drafting curriculum, he took a job as an AutoCAD drafter/designer at a naval architecture firm in Portsmouth, Virginia. It was there he was introduced to AutoLISP.

For the next few years, Phill wrote code exclusively in AutoLISP. He began dabbling with Visual Lisp shortly after its inception and, at about the same time, started writing macros for Excel in Visual Basic for Applications. When Autodesk included VBA in AutoCAD, everything came together.

Eight years later, Phill is a senior developer of ShipWorks, the only shipbuilding AutoCAD add-on used by Northrop Grumman Newport News in the design of next-generation aircraft carriers.

Phill lives in Virginia Beach, Virginia, with his wife, Amy; sons, Alex and Rowan; and daughter, Morgan.

STEVE JOHNSON was born in England but has lived in Perth, Western Australia, since 1984. He is married with two children and has a bachelor's degree in applied science (information science). Steve has been an AutoCAD specialist since 1985. His company, cad nauseam, provides AutoCAD-based consulting, development, support, training, and technical writing services. His software for AutoCAD is used by hundreds of clients around the world.

Steve is a contributing editor of *Cadalyst* magazine and has written the monthly column *Bug Watch* since 1995. He is also the vice president, Asia Pacific, of Cadlock and a past president of the Western Australian AutoCAD User Group (WAAUG). As a former committee member on the Applications Programming Special Interest Group of the North American Autodesk User Group, he contributed to the NAAUG newsletter. NAAUG is now known as Autodesk User Group International (AUGI). Steve was part of the AUGI Benchmark Committee, which assisted in the development of the AUGI Gauge, a comprehensive AutoCAD benchmark.

Acknowledgments

I have many, many people to thank on this project. Please forgive me if I miss mentioning your name. First, the people at Apress: without them I would not have had the opportunity to create this work. Thanks to Gary Cornell for his initial acceptance and input, Kylie Johnston for coordinating things, and Tony Davis for getting it all started.

A great big thank you to the editorial and production staff: Candace English and Kim Wimpsett, my copy editors; Janet Vail, production editor; Nancy Wright, formatter; Linda Weidemann, compositor; Linda Seifert, proofreader; and Kevin Broccoli, indexer. I enjoyed working with each of you.

Second, a special thanks to my technical reviewers, Phillip Ash and Steve Johnson.

A special thank you to my children—Stephen, Clair, David, and Emily—for understanding when Daddy had to work.

Lastly, thanks to those who helped and whose names I forgot to include here. To everyone in the AutoCAD community with whom I've had the pleasure of crossing paths, from the bottom of my heart, thanks!

Introduction

This book provides a concise guide to the kind of customization programmers can achieve with AutoCAD 2006. It demonstrates how to use AutoCAD through short code examples written in Visual Basic for Applications (VBA). It also includes a complete quick reference that lists all the events, methods, and properties available with AutoCAD. Finally, it describes all the constants and system variables.

What Is AutoCAD?

So, what is AutoCAD? First released in 1982 under the name MicroCAD, AutoCAD has become a powerful tool for drafting and design purposes. AutoCAD 2006 incorporates many new features to enhance flexibility and drawing control. To reflect this extra functionality, many new ActiveX objects, properties, methods, and events have been included for improved programmability.

What Is This Book About?

This book is about AutoCAD 2006 and how to use AutoCAD VBA in your applications to handle all your drawing tasks more efficiently. It shows you how to programmatically control the creation and editing of individual drawing objects, manipulate linetypes and layers, control text and dimension styles, and do much more. As you encounter each of these topics, you'll learn all about the associated objects, including their properties, methods, and events.

By interfacing with AutoCAD, you can exploit all of AutoCAD's functionality that would have taken you a long time to write yourself. This book will first help you learn how to use this functionality. Then it will become a handy reference later, when you have a question that you just can't answer.

This book splits topics into neat and intuitive segments and makes it easy to find specific information when you need it (that is, when you're coding real-world applications).

This book is divided into three main parts:

- Chapters 1 through 3 provide a rapid introduction to Visual Basic and explain the notation and commands particular to AutoCAD VBA projects.
- Chapters 4 through 22 supply a detailed breakdown of most of the AutoCAD object model, covering common tasks complete with several varied code examples demonstrating how to use the relevant objects' methods and properties.
- Finally, the quick-reference appendixes describe all the members of all the AutoCAD objects at a glance. Appendix D, on Object Model Cross-Reference, can be found on the Apress website (`www.apress.com`).

Who Is This Book For?

The book is a reference guide for AutoCAD programmers, and it's primarily designed to explain and demonstrate the features of AutoCAD 2006. As such, this isn't a beginner's guide; however, if you've programmed in any language that can interface with other COM objects, you should be able to easily understand and use this book.

In particular, the book is aimed at programmers who use AutoCAD for daily tasks and can see the benefits of customizing and automating these tasks. I present programming techniques needed to create and modify AutoCAD drawings, customize preferences, query and set system variables, and so on, using the built-in VBA.

You can customize AutoCAD to any degree of sophistication. If you can think it up, then I bet you can use AutoCAD VBA and this book to help you achieve your goal.

Tell Us What You Think

I've worked hard on this book to make it enjoyable and useful. My best reward would be to hear from you that you liked it and that it was worth the money you paid for it. I've done my best to try to understand and match your expectations.

Apress and I would like to know what you think about it. Tell us what you liked best, and what we could have done better. If you think this is just a marketing gimmick, then test us—drop us a line! We'll answer, and we'll take whatever you say under consideration for future editions. The easiest way to do so is to send e-mail to `feedback@apress.com`.

You can also find more details about Apress on the Web site at `http://www.apress.com`. There you'll find the code from the latest Apress books, sneak previews of forthcoming titles, and information about the authors and editors. You can order Apress titles directly from the site or find out where your nearest local bookstore with Apress titles is located.

Customer Support

If you find a mistake in the book, your first port of call should be the errata page for this book on the Web site at `http://www.apress.com`. You can see any errata already posted there or submit your own.

If you can't find an answer there, send an e-mail to `support@apress.com` telling us about the problem. We'll do everything we can to answer promptly. Please remember to let us know the book title your query relates to and, if possible, the page number. This will help us reply to you quickly.

CHAPTER 1

The VBA Integrated Development Environment (VBAIDE)

Within AutoCAD, you develop VBA programs in the Visual Basic for Applications (VBA) Integrated Development Environment (IDE). As it does with the Visual LISP IDE, Autodesk provides the VBAIDE as an integral part of many of its products, including AutoCAD. Unlike the Visual LISP IDE, however, Microsoft licenses the VBAIDE to Autodesk for inclusion in its products. Therefore, its features are from Microsoft, not Autodesk.

This chapter explores the VBAIDE environment's facets and shows you how to take advantage of its tools. It covers these topics:

- Visual Basic concepts
- Starting the editor
- Exploring the user interface
- Managing projects
- Using the text editor
- The Object Browser

Visual Basic Concepts

Since VBA is a Microsoft Windows development environment, you'll find developing in VBA easiest if you have some knowledge of Windows. If you are new to Windows, you'll find fundamental differences between programming in Windows and programming in other environments, such as Visual LISP. The next sections outline concepts of Windows programming that may be new to you.

Windows, Events, and Messages

Explaining the inner workings of Windows requires much more space than is available in this book. But you don't need an extensive knowledge of the Windows workings to create useful

applications. The Windows operating system can be simplified to three basic concepts: windows, events, and messages.

A window is a rectangular region on the screen that has its own border. The AutoCAD drawing window, Notepad, a Word document, and the place where you compose an e-mail are all windows.

Windows can have hierarchy. A dialog form, which is a window within a distinct application, contains relevant ActiveX components and code. A drawing window is the parent of a dialog form window, and AutoCAD is the parent of the drawing windows within it. The operating system is the parent "window" of all applications running in it, including AutoCAD.

Each window recognizes and controls the programs that execute inside them or their subordinate windows. To manage windows, Windows assigns a unique ID known as a *handle*, or *hWnd* in programming jargon, to each window. Windows uses *events* to constantly monitor each window for signs of activity. Events change the application environment or *system state*. They occur when a user acts, such as by clicking the mouse or pressing a key; programmatically; or by another window through system processes.

Each time an event is triggered, Windows sends a message to the hosting application. Windows processes the message and broadcasts it to the windows. Then, based on its own instructions, each window can take appropriate action such as repainting itself when uncovered by another window. In the case of VBA in AutoCAD, the VBA work space intercepts event messages. VBA programs can then respond directly or pass the event up to AutoCAD or to Windows if necessary. VBA provides a controlling environment within AutoCAD in which to execute and respond to events, either directly or by allowing AutoCAD or Windows to respond.

Although this seems like a lot of work, VBA hides most of the low-level details from you and exposes *event procedures*, which are routines that execute when a particular event occurs, for your convenience. You can quickly create very powerful applications without being concerned with low-level details.

Event-Driven vs. Procedural Programming

When a traditional procedural application runs, it follows a predetermined path that controls the portions and sequence of code executed. It starts with the first line of code and progresses from the top down, calling each procedure when needed, until reaching the end of the code. This predetermined path is the major difference between procedural and event-driven applications.

Event-driven applications do not have a predetermined destiny. Different sections of code are executed based upon the events triggered in whatever order they occur. Depending on what events occur when the application runs, some sections of code may not get executed at all.

You can't predict the sequence of events, so you must make assumptions about the application's "state" at any moment. This seems like it would be difficult, but it's really not. Typically, you have a set of possibilities to work with, such as the `Click`, `DblClick`, `KeyPress`, and `LostFocus` events. For example, you might require the user to type a value in a `TextBox` before enabling a `CommandButton` that allows further processing. The `TextBox` control's `Change` event would contain code that enables the `CommandButton` control, as shown in this sample code:

```
Private Sub TextBox1_Change()
  If Len(TextBox1.Text) > 0 Then
    CommandButton1.Enabled = True

    Else
      CommandButton1.Enabled = False
  End If
End Sub
```

Each time you add or delete text from the `TextBox` control, the program executes this event procedure. The code checks the length of the text and if it is greater than zero, meaning there is something in the `TextBox`, it enables the `CommandButton`. Otherwise, it disables the `CommandButton`.

Programmatically changing the text in the `TextBox` triggers the `Change` event. If you allow for this occurrence, you might get unexpected results. Using events is very powerful, but you must know what each event might trigger elsewhere in your application.

Developing Your Applications Interactively

In more-traditional development environments, most developers follow a distinct three-step process: writing, compiling, and testing. However, VBA uses a more interactive approach to development that makes it easier for both beginning and experienced developers.

Languages such as C++ require you to write all the code then compile it. During the compile, you may uncover numerous errors, from simple typing errors to more-complex syntax errors. The Visual Basic programming environment, on the other hand, interprets your code every line of the way, alerting you to potential problems now instead of during a lengthy compile cycle.

Because Visual Basic is partially compiling your code as you type it, it takes very little time to finish compiling the code and execute your application. Unlike with other languages, you will find that you are constantly writing, executing, and refining your application. In addition, the Visual Basic environment employs a graphical environment. You most often work in the graphical interface first and on the code second. This lets you spend more time creating and less time compiling and recompiling.

Starting the Editor

One of the first questions that you will face is "How do I enter source code (structured commands) or develop a user interface (forms or dialog boxes)?" The answer is the IDE, a graphical user interface you use to develop applications. It is similar to development environments provided in other applications, such as Microsoft Access and Microsoft Excel.

To display the VBAIDE, choose Tools ➤ Macro ➤ Visual Basic Editor or press Alt+F11. Alternatively, you can start the editor by typing VBAIDE at the AutoCAD command prompt, as shown in Figure 1-1.

Figure 1-1. *The AutoCAD command prompt*

■ **Note** AutoCAD includes a Visual Basic menu and toolbar. You can load it by copying `ACAD.DVB` from \Sample\VBA\VBAIDEMenu (in the AutoCAD folder) to a directory in your support path. The toolbar will then autoload when the first VB command is launched. If you already have an `ACAD.DVB`, you can cut and paste from the provided `ACAD.DVB` to your own. This provides an AutoCAD toolbar for the following commands: VBAIDE, VBAPREF, VBAMAN, VBALOAD, and VBARUN. This also adds New, Open, and Close to the File pull-down on the VBAIDE toolbar.

Exploring the User Interface

The editor is composed of several different windows. The first time you open it, it looks like Figure 1-2. Use the View menu to control which windows are visible. To get context-sensitive help on any window, click in it and press F1.

The rest of this section discusses the most frequently used windows.

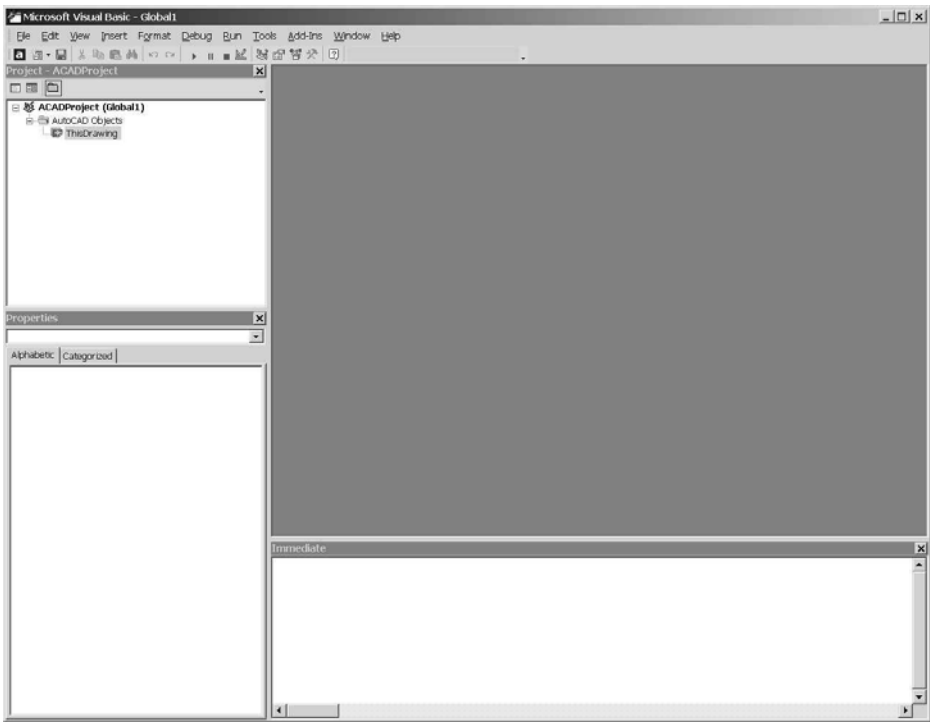

Figure 1-2. *The AutoCAD VBAIDE*