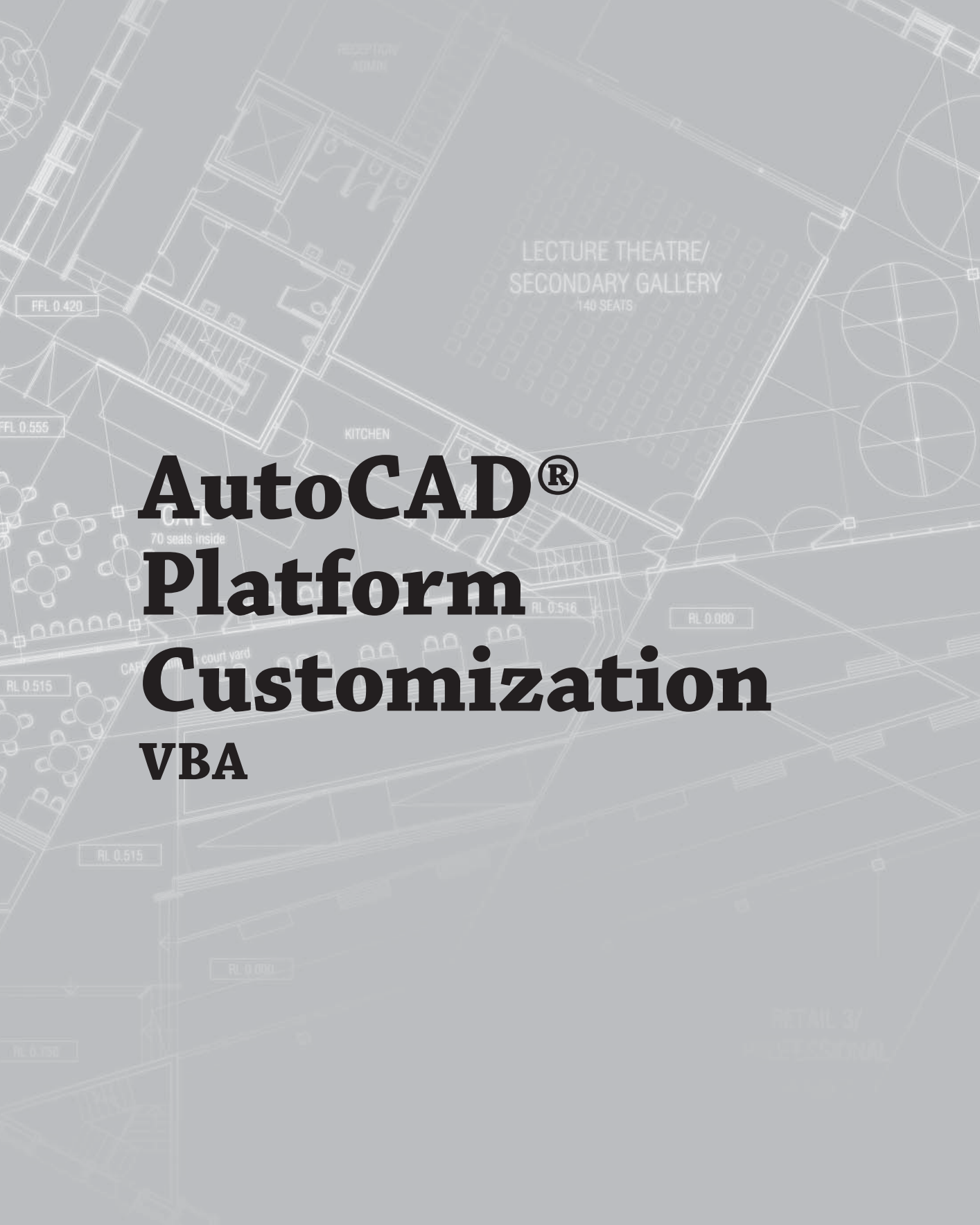

LEE AMBROSIUS
LECTURE THEATRE/
SECONDARY GALLERY
140 SEATS
FFL 0.420
FFL 0.555
KITCHEN
CAFE
70 seats inside
RL 0.516
RL 0.000
CAFE seating in court yard
RL 0.515
AutoCAD® Platform Customization
VBA
SYBEX
A Wiley Brand



AutoCAD® Platform Customization

VBA

AutoCAD® Platform Customization

VBA

Lee Ambrosius

Autodesk®
Official Training Guide

SYBEX®
A Wiley Brand

Senior Acquisitions Editor: Stephanie McComb
Development Editor: Mary Ellen Schutz
Technical Editor: Richard Lawrence
Production Editor: Dassi Zeidel
Copy Editor: Liz Welch
Editorial Manager: Pete Gaughan
Production Manager: Kathleen Wisor
Associate Publisher: Jim Minatel
Book Designers: Maureen Forys, Happenstance Type-O-Rama; Judy Fung
Proofreader: Candace Cunningham
Indexer: Ted Laux
Project Coordinator, Cover: Brent Savage
Cover Designer: Wiley
Cover Image: © Smileyjoanne/iStockphoto.com

ISBN: 978-1-118-90044-4 (ebk.)
ISBN: 978-1-118-90698-9 (ebk.)

Library of Congress Control Number: 2015936845

10 9 8 7 6 5 4 3 2 1

To my friend Kathy Enderby: You were one of the first people to encourage me to follow my passion for programming and sharing what I had learned with others. Thank you for believing in me all those years ago and for being there when I needed someone to bounce ideas off—especially during those late-night scrambles right before deploying a new software release.

Acknowledgments

I have to give a very special thanks to all the great folks at Sybex, especially Willem Knibbe, for working on and helping to get this project off the ground after a few years of talking about it. The next two people I would like to thank are Mary Ellen Schutz and Dassi Zeidel, the development and production editors on this book; you two made sure I stayed on track and delivered a high-quality book. I also want to thank Liz Welch (copyeditor), Candace Cunningham (proofreader), and Ted Laux (indexer) for the work you all did on this book.

Thanks to all the folks at Autodesk, who put in the long hours and are dedicated to the work they do on the Autodesk® AutoCAD® product. I cannot forget one of the most important individuals on this book, my technical editor, Richard Lawrence. Richard is a great friend who I met many years ago at Autodesk University. He is a passionate and driven user of AutoCAD and is always looking to improve the way he uses AutoCAD. Richard, I appreciate everything that you have done to make this book better. Congrats on making it through your first book as a technical editor.

About the Author

Lee Ambrosius first started working with AutoCAD R12 for DOS in 1994. As a drafter, he quickly discovered that every project included lots of repetition. Lee, not being one to settle for "this is just the way things are," set out on a path that would redefine his career. This new path would lead him into the wondrous world of customization and programming—which you might catch him referring to as "the rabbit hole."

In 1996, Lee began learning the core concepts of customizing the AutoCAD user interface and AutoLISP. The introduction of VBA in AutoCAD R14 would once again redefine how Lee approached programming solutions for AutoCAD. VBA made it much easier to communicate with external databases and other applications that supported VBA. It transformed the way information could be moved between project-management and manufacturing systems.

Not being content with VBA, in 1999 Lee attended his first Autodesk University and began to learn ObjectARX®. Autodesk University had a lasting impression on him. In 2001, he started helping as a lab assistant. He began presenting on customizing and programming AutoCAD at the event in 2004. Along the way he learned how to use the AutoCAD Managed .NET API.

In 2005, Lee decided cubicle life was no longer for him, so he ventured off into the CAD industry as an independent consultant and programmer with his own company, HyperPics, LLC. After he spent a few years as a consultant, Autodesk invited him to work on the AutoCAD team; he has been on the AutoCAD team since 2007. For most of his career at Autodesk, Lee has worked primarily on customization and end-user documentation. Recently, he has been working on the AutoLISP, VBA, ObjectARX, .NET, and JavaScript programming documentation.

In addition to working on the AutoCAD documentation, Lee has been involved as a technical editor or author for various editions of the *AutoCAD and AutoCAD LT Bible*, *AutoCAD for Dummies*, *AutoCAD & AutoCAD LT All-in-One Desk Reference for Dummies*, *AutoCAD 3D Modeling Workbook for Dummies*, and *Mastering AutoCAD for Mac.* He has also written white papers on customization for Autodesk and a variety of articles on customization and programming for *AUGIWorld*, published by AUGI®.

Contents at a Glance

Contents

Introduction

Welcome to *AutoCAD Platform Customization: VBA*! Have you ever thought to yourself, why doesn't the Autodesk® AutoCAD® program include every feature I need? Why isn't it streamlined for the type of work I perform? If so, you are not alone. AutoCAD at its core is a drafting platform that, through programming, can be shaped and molded to more efficiently complete the tasks you perform on a daily basis and enhance your company's workflows. Take a deep breath. I did just mention programming, but programming isn't something to fear. At first, just the idea of programming makes many people want to run in the opposite direction—myself included. The productivity gains are what propelled me forward. Programming isn't all that different from anything else you've tried doing for the first time.

In many ways, learning to program is much like learning a foreign language. For many new to Visual Basic for Applications (VBA), the starting place is learning the basics: the syntax of the programming language and how to leverage commands and system variables. Executing commands and working with system variables using the `SendCommand` and `PostCommand` methods can be a quick way to get started and become comfortable with VBA. After you are comfortable with the syntax of VBA and the `SendCommand` and `PostCommand` functions, you can begin to learn how to access the AutoCAD Object library to develop more complex and robust programs.

About This Book

AutoCAD Platform Customization: VBA provides you with an understanding of the VBA programming language and how it can be used in combination with the AutoCAD Object library to improve your productivity. This book is designed to be more than just an introduction to VBA and the AutoCAD Object library; it is a resource that can be used time and again when developing VBA programs for use with AutoCAD. As you page through this book, you will notice that it contains sample code and exercises that are based on real-world solutions.

This book is the third and final book in a series that focuses on customizing and programming AutoCAD. The three-book series as a whole is known as *AutoCAD Platform Customization: User Interface, AutoLISP, VBA, and Beyond*, which will be available as a printed book in 2015. Book 1 in the series, *AutoCAD Platform Customization: User Interface and Beyond*, was published in early 2014 and focused on CAD standards and general customization of AutoCAD; Book 2, *AutoCAD Platform Customization: AutoLISP*, was published in mid-2014 and covers the AutoLISP programming language.

Is This Book for You?

AutoCAD Platform Customization: VBA covers many aspects of VBA programming for AutoCAD on Windows. If any of the following are true, this book will be useful to you:

- You want to develop and load custom programs with the VBA programming language for use in the AutoCAD drawing environment.
- You want to automate the creation and manipulation of drawing objects.
- You want to automate repetitive tasks.
- You want to help manage and enforce CAD standards for your company.

NOTE VBA programming isn't supported for AutoCAD on Mac OS.

VBA in AutoCAD

VBA is often overlooked as one of the options available to extend the AutoCAD program. There is no additional software to purchase, but you must download and install a release-specific secondary component to use VBA. You can leverage VBA to perform simple tasks, such as inserting a title block with a specific insertion point, scale, and rotation and placing the block reference on a specific layer. To perform the same tasks manually, end users would have to first set a layer as current, choose the block they want to insert, and specify the properties of the block, which in the case of a title block are almost always the same.

The VBA programming language and AutoCAD Object library can be used to do the following:

- Create and manipulate graphical objects in a drawing, such as lines, circles, and arcs
- Create and manipulate nongraphical objects in a drawing, such as layers, dimension styles, and named views
- Perform mathematical and geometric calculations
- Request input from or display messages to the user at the Command prompt
- Interact with files and directories in the operating system
- Read from and write to external files
- Connect to applications that support ActiveX and COM
- Display user forms and get input from the end user

VBA code statements are entered into the Visual Basic Editor and stored in a DVB file. Once a VBA project has been loaded, you can execute the macros through the Macros dialog box. Unlike standard AutoCAD commands, macros cannot be executed from the Command prompt, but once executed, a macro can prompt users for values at the Command prompt or with a user form. It is possible to execute a macro from a command macro that is activated with a command button displayed in the AutoCAD user interface or as a tool on a tool palette.

What to Expect

This book is organized to help you learn VBA fundamentals and how to use the objects in the AutoCAD Object library. Additional resources and files containing the example code found throughout this book can be found on the companion web page, `www.sybex.com/go/autocadcustomization`.

Chapter 1: Understanding the AutoCAD VBA Environment In this chapter, you'll get an introduction to the Visual Basic Editor. I begin by showing you how to verify whether the VBA environment for AutoCAD has been installed and, if not, how to install it. After that, you are eased into navigating the VBA Editor and managing VBA programs. The chapter wraps up with learning how to execute macros and access the help documentation.

Chapter 2: Understanding the Visual Basic for Application In this chapter, you'll learn the fundamentals of the VBA programming language and how to work with objects. VBA fundamentals include a look at the syntax and structure of a statement, how to use a function, and how to work with variables. Beyond syntax and variables, you learn to group multiple statements into a custom procedure.

Chapter 3: Interacting with the Application and Documents Objects In this chapter, you'll learn to work with the AutoCAD application and manage documents. Many of the tasks you perform with an AutoCAD VBA program require you to work with either the application or a document. For example, you can get the objects in a drawing and even access end-user preferences. Although you typically work with the current document, VBA allows you to work with all open documents and create new documents. From the current document, you can execute commands and work with system variables from within a VBA program, which allows you to leverage and apply your knowledge of working with commands and system variables.

Chapter 4: Creating and Modifying Drawing Objects In this chapter, you'll learn to create and modify graphical objects in model space with VBA. Graphical objects represent the drawing objects, such as a line, an arc, or a circle. The methods and properties of an object are used to modify and obtain information about the object. When working with the objects in a drawing, you can get a single object or step through all objects in a drawing.

Chapter 5: Interacting with the User and Controlling the Current View In this chapter, you'll learn to request input from an end user and manipulate the current view of a drawing. Based on the values provided by the end user, you can then determine the end result of the program. You can evaluate the objects created or consider how a drawing will be output and use that information to create named views and adjust the current view in which objects are displayed.

Chapter 6: Annotating Objects In this chapter, you'll learn how to create and modify annotation objects. Typically, annotation objects are not part of the final product that is built or manufactured based on the design in the drawing. Rather, annotation objects are used to communicate the features and measurements of a design. Annotation can be a single line of text that is used as a callout for a leader, a dimension that indicates the distance between two

drill holes, or a table that contains quantities and information about the windows and doors in a design.

Chapter 7: Working with Blocks and External References In this chapter, you'll learn how to create, modify, and manage block definitions. Model space in a drawing is a special named block definition, so working with block definitions will feel familiar. Once you create a block definition, you will learn how to insert a block reference and work with attributes along with dynamic properties. You'll complete the chapter by learning how to work with externally referenced files.

Chapter 8: Outputting Drawings In this chapter, you will learn how to output the graphical objects in model space or on a named layout to a printer, plotter, or electronic file. Named layouts will be used to organize graphical objects for output, including title blocks, annotation, floating viewports, and many others. Floating viewports will be used to control the display of objects from model space on a layout at a specific scale. After you define and configure a layout, you learn to plot and preview a layout. The chapter wraps up with covering how to export and import file formats.

Chapter 9: Storing and Retrieving Custom Data In this chapter, you will learn how to store custom information in a drawing or in the Windows Registry. Using extended data (Xdata), you will be able to store information that can be used to identify a graphical object created by your program or define a link to a record in an external database. In addition to attaching information to an object, you can store data in a custom dictionary that isn't attached to a specific graphical object in a drawing. Both Xdata and custom dictionaries can be helpful in making information available between drawing sessions; the Windows Registry can persist data between sessions.

Chapter 10: Modifying the Application and Working with Events In this chapter, you will learn how to customize and manipulate the AutoCAD user interface. You'll also learn how to load and access externally defined custom programs and work with events. Events allow you to respond to an action that is performed by the end user or the AutoCAD application. There are three main types of events that you can respond to: application, document, and object.

Chapter 11: Creating and Displaying User Forms In this chapter, you will learn how to create and display user forms. User forms provide a more visual approach to requesting input from the user.

Chapter 12: Communicating with Other Applications In this chapter, you will learn how to work with libraries provided by other applications. These libraries can be used to access features of the Windows operating system, read and write content in an external text or XML file, and even work with the applications that make up Microsoft Office.

Chapter 13: Handling Errors and Deploying VBA Projects In this chapter, you will learn how to catch and handle errors that are caused by the incorrect use of a function or the improper handling of a value that is returned by a function. The Visual Basic Editor provides tools that allow you to debug code statements, evaluate values assigned to user-defined variables, identify where within a program an error has occurred, and determine how errors should be handled. The chapter wraps everything up with covering how to deploy a VBA project on other workstations for use by individuals at your company.

Bonus Chapter 1: Working with 2D Objects and Object Properties In this chapter, you build on the concepts covered in Chapter 4, "Creating and Modifying Drawing Objects." You will learn to create additional types of 2D objects and use advanced methods of modifying objects; you also learn to work with complex 2D objects such as regions and hatch fills. The management of layers and linetypes and the control of the appearance of objects are also covered.

Bonus Chapter 2: Modeling in 3D Space In this chapter, you learn to work with objects in 3D space and 3D with objects. 3D objects can be used to create a model of a drawing which can be used to help visualize a design or detect potential design problems. 3D objects can be viewed from different angles and used to generate 2D views of a model that can be used to create assembly directions or shop drawings.

Bonus Chapter 3: Development Resources In this chapter, you discover resources that can help expand the skills you develop from this book or locate an answer to a problem you might encounter. I cover development resources, as well as places you might be able to obtain instructor-led training and interact with fellow users on extending AutoCAD. The online resources listed cover general customization, AutoLISP, and VBA programming in AutoCAD.

NOTE Bonus Chapter 1, Bonus Chapter 2, and Bonus Chapter 3 are located on the companion website.

Companion Website

An online counterpart to this book, the companion website contains the sample files required to complete the exercises found in this book, in addition to the sample code and project files used to demonstrate some of the programming concepts explained in this book. In addition to the sample files and code, the website contains resources that are not mentioned in this book, such as the bonus chapters. The companion website can be found at `www.sybex.com/go/autocadcustomization`.

Other Information

This book assumes that you know the basics of your operating system and AutoCAD 2009 or later. When appropriate, I indicate when a feature does not apply to a specific operating system or release of AutoCAD. Most of the images in this book were taken using AutoCAD 2014 in Windows 8.

Neither AutoCAD LT® nor AutoCAD running on Mac OS support the VBA programming platform, none of the content in this book can be used if you are working on Mac OS.

Styles and Conventions of This Book

This book uses a number of styles and character formats—bold, italic, monotype face, and all uppercase or lowercase letters, among others—to help you distinguish between the text you read, sample code you can try, text that you need to enter at the AutoCAD Command prompt, or the name of an object class or method in one of the programming languages.

As you read through this book, keep the following conventions in mind:

- User-interface selections are represented by one of the following methods:
  - Click the Application button ➢ Options.
  - On the ribbon, click the Manage tab ➢ Customization ➢ User Interface.
  - On the menu bar, click Tools ➢ Customize ➢ Interface.
  - In the drawing window, right-click and click Options.
- Keyboard input is shown in bold (for example, type **`cui`** and press Enter).
- Prompts that are displayed at the AutoCAD Command prompt are displayed as monospace font (for example, `Specify a start point:`).
- AutoCAD command and system variable names are displayed in all lowercase letters with a monospace font (for example, `line` or `clayer`).
- VBA function and AutoCAD Object library member names are displayed in mixed-case letters with a monospace font (for example, `Length` or `SendCommand`).
- Example code and code statements that appear within a paragraph are displayed in monospace font. Code might look like one of the following:
  - `MsgBox "ObjectName: " & oFirstEnt.ObjectName`
  - The `MsgBox` method can be used to display a text message to the user
  - `' Gets the first object in model space`

Contacting the Author

I hope that you enjoy *AutoCAD Platform Customization: VBA* and that it changes the way you think about completing your day-to-day work. If you have any feedback or ideas that could improve this book, you can contact me using the following address:

Lee Ambrosius: `lee_ambrosius@hyperpics.com`

On my blog and website, you'll find additional articles on customization and samples that I have written over the years. You'll find these resources here:

Beyond the UI: `http://hyperpics.blogs.com`

HyperPics: `www.hyperpics.com`

If you encounter any problems with this publication, please report them to the publisher. Visit the book's website, `www.sybex.com/go/autocadcustomization`, and click the Errata link to open a form and submit the problem you found.

Chapter 1

Understanding the AutoCAD VBA Environment

More than 15 years ago, Visual Basic (VB) was the first modern programming language I learned. This knowledge was critical to taking my custom programs to a whole new level. VB allows you to develop stand-alone applications that can communicate with other programs using Microsoft's Object Linking and Embedding (OLE) and ActiveX technologies. Autodesk® AutoCAD® supports a variant of VB known as Visual Basic for Applications (VBA) that requires a host application to execute the programs you write; it can't be used to develop stand-alone executable files.

I found VB easier to learn than AutoLISP® for a couple of reasons. First, there are, in general, many more books and online resources dedicated to VB. Second, VB syntax feels more natural. By natural, I mean that VB syntax reads as if you are explaining a process to someone in your own words, and it doesn't contain lots of special characters and formatting like many other programming languages.

As with learning anything new, there will be a bit of hesitation on your part as you approach your first projects. This chapter eases you into the AutoCAD VBA environment and the VB programming language.

What Makes Up an AutoCAD VBA Project?

Custom programs developed with VBA implemented in the AutoCAD program are stored in a project that has a `.dvb` file extension. VBA projects contain various objects that define a custom program. These objects include the following:

- Code modules that store the custom procedures and functions that define the primary functionality of a custom program
- `UserForms` that define the dialog boxes to be displayed by a custom program
- Class modules that store the definition of a custom object for use in a custom program
- Program library references that contain the dependencies a custom program relies on to define some or all of the functionality

The AutoCAD VBA Editor is an integrated development environment (IDE) that allows for the creation and execution of macros stored in a project file. A macro is a named block of code that can be executed from the AutoCAD user interface or privately used within a project. You can also enter and execute a single VBA statement at the AutoCAD Command prompt using the `vbastmt` command.

The most recent generation of VB is known as VB.NET. Although VB and VB.NET have similar syntax, they are not the same. VBA, whether in AutoCAD or other programs such as Microsoft Word, is based on VB6 and not VB.NET. If you are looking for general information on VBA, search the Internet using the keywords VBA and VB6.

Real World Scenario

If You Have Conversations Like This, You Can Code Like This

The summer intern had one job—add a layer and a confidentiality note to a series of 260 production drawings. September arrived, the intern left for school, and now your manager is in your cubicle.

"Half of these drawings are missing that confidentiality note Purchasing asked for. I need you to **add** that new **layer**, name it **Disclaimer**, and then **add** the confidentiality note as **multiline text** to **model space**. The note should be located on the new **Disclaimer** layer at **0.25,0.1.75,0** with a **height** of **0.5,** and the text should read **Confidential: This drawing is for use by internal employees and approved vendors only**. Be sure to check to see **if paper space** is **active.** If it is, **then** set **model space active** per the new standards before you **save** each drawing," he says.

"I can do that," you respond.

"Can you manage it by close of day tomorrow? The parts are supposed to go out for quote on Wednesday morning."

"Sure," you tell him, knowing that a few lines of VBA code will allow you to make the changes quickly.

So, you sit down and start to code. The conversation-to-code translation flows smoothly. (Notice how many of the words in the conversation flow right into the actual VBA syntax.)

```
With ThisDrawing
  .Layers.Add "Disclaimer"

  Dim objMText As AcadMText

  Dim insPt(2) As Double
  insPt(0) = 0.25: insPt(1) = 1.75: insPt(2) = 0

  Set objMText = .ModelSpace.AddMText(insPt, 15, _
    "Confidential: This drawing is for use by internal " & _
    "employees and approved vendors only")

  objMText.Layer = "Disclaimer"
  objMText.Height = 0.5

  If .ActiveSpace = acPaperSpace Then
    .ActiveSpace = acModelSpace
   End If

  .Save
End With
```