

LEE AMBROSIUS

LECTURE THEATRE/  
SECONDARY GALLERY  
140 SEATS

KITCHEN

CAFE

70 seats inside

CAFE seating in court yard

# AutoCAD® Platform Customization

VBA



**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 Forsy, 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

Copyright © 2015 by John Wiley & Sons, Inc., Indianapolis, Indiana  
Published simultaneously in Canada

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

No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, except as permitted under Sections 107 or 108 of the 1976 United States Copyright Act, without either the prior written permission of the Publisher, or authorization through payment of the appropriate per-copy fee to the Copyright Clearance Center, 222 Rosewood Drive, Danvers, MA 01923, (978) 750-8400, fax (978) 646-8600. Requests to the Publisher for permission should be addressed to the Permissions Department, John Wiley & Sons, Inc., 111 River Street, Hoboken, NJ 07030, (201) 748-6011, fax (201) 748-6008, or online at <http://www.wiley.com/go/permissions>.

**Limit of Liability/Disclaimer of Warranty:** The publisher and the author make no representations or warranties with respect to the accuracy or completeness of the contents of this work and specifically disclaim all warranties, including without limitation warranties of fitness for a particular purpose. No warranty may be created or extended by sales or promotional materials. The advice and strategies contained herein may not be suitable for every situation. This work is sold with the understanding that the publisher is not engaged in rendering legal, accounting, or other professional services. If professional assistance is required, the services of a competent professional person should be sought. Neither the publisher nor the author shall be liable for damages arising herefrom. The fact that an organization or Web site is referred to in this work as a citation and/or a potential source of further information does not mean that the author or the publisher endorses the information the organization or Web site may provide or recommendations it may make. Further, readers should be aware that Internet Web sites listed in this work may have changed or disappeared between when this work was written and when it is read.

For general information on our other products and services or to obtain technical support, please contact our Customer Care Department within the U.S. at (877) 762-2974, outside the U.S. at (317) 572-3993 or fax (317) 572-4002.

Wiley publishes in a variety of print and electronic formats and by print-on-demand. Some material included with standard print versions of this book may not be included in e-books or in print-on-demand. If this book refers to media such as a CD or DVD that is not included in the version you purchased, you may download this material at <http://booksupport.wiley.com>. For more information about Wiley products, visit [www.wiley.com](http://www.wiley.com).

**Library of Congress Control Number:** 2015936845

**TRADEMARKS:** Wiley, the Wiley logo, and the Sybex logo are trademarks or registered trademarks of John Wiley & Sons, Inc. and/or its affiliates, in the United States and other countries, and may not be used without written permission. AutoCAD is a registered trademark of Autodesk, Inc. All other trademarks are the property of their respective owners. John Wiley & Sons, Inc. is not associated with any product or vendor mentioned in this book.

10987654321

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

<i>Introduction</i> .....	<i>xxi</i>
Chapter 1 • Understanding the AutoCAD VBA Environment .....	1
Chapter 2 • Understanding Visual Basic for Applications .....	21
Chapter 3 • Interacting with the Application and Documents Objects .....	57
Chapter 4 • Creating and Modifying Drawing Objects .....	83
Chapter 5 • Interacting with the User and Controlling the Current View.....	113
Chapter 6 • Annotating Objects .....	151
Chapter 7 • Working with Blocks and External References .....	175
Chapter 8 • Outputting Drawings .....	221
Chapter 9 • Storing and Retrieving Custom Data .....	247
Chapter 10 • Modifying the Application and Working with Events.....	279
Chapter 11 • Creating and Displaying User Forms .....	309
Chapter 12 • Communicating with Other Applications .....	339
Chapter 13 • Handling Errors and Deploying VBA Projects .....	375
<i>Index</i> .....	<i>409</i>



# Contents

<i>Introduction</i> .....	<i>xxi</i>
---------------------------	------------

## **Chapter 1 • Understanding the AutoCAD VBA Environment ..... 1**

What Makes Up an AutoCAD VBA Project? .....	1
What You'll Need to Start .....	3
Determine If the AutoCAD VBA Environment Is Installed .....	3
Install the AutoCAD 2015 VBA Enabler .....	4
Getting Started with the VBA Editor .....	4
Identifying the Components of a VBA Project .....	5
Navigating the VBA Editor Interface .....	7
Setting the VBA Environment Options .....	11
Managing VBA Programs .....	11
Creating a New VBA Project .....	12
Saving a VBA Project .....	13
Loading and Unloading a VBA Project .....	13
Embedding or Extracting a VBA Project .....	15
Executing VBA Macros .....	16
Accessing the AutoCAD VBA Documentation .....	19

## **Chapter 2 • Understanding Visual Basic for Applications ..... 21**

Learning the Fundamentals of the VBA Language .....	21
Creating a Procedure .....	22
Declaring and Using Variables .....	24
Controlling the Scope of a Procedure or Variable .....	26
Continuing Long Statements .....	27
Adding Comments .....	28
Understanding the Differences Between VBA 32- and 64-Bit .....	29
Exploring Data Types .....	30
Working with Objects .....	32
Accessing Objects in a Collection .....	34
Storing Data in Arrays .....	35
Calculating Values with Math Functions and Operators .....	38
Manipulating Strings .....	39
Converting Between Data Types .....	42
Comparing Values .....	44
Testing Values for Equality .....	44
Comparing String Values .....	45
Determining If a Value Is Greater or Less Than Another .....	46

Checking for Null, Empty, or Nothing Values . . . . .	47
Validating Values . . . . .	48
Grouping Comparisons . . . . .	48
Conditionalizing and Branching Statements . . . . .	49
Evaluating If a Condition Is Met . . . . .	49
Testing Multiple Conditions . . . . .	51
Repeating and Looping Expressions . . . . .	52
Repeating Expressions a Set Number of Times . . . . .	52
Stepping Through an Array or Collection . . . . .	53
Performing a Task While or Until a Condition Is Met . . . . .	54
<b>Chapter 3 • Interacting with the Application and Documents Objects . . .</b>	<b>57</b>
Working with the Application . . . . .	57
Getting Information about the Current AutoCAD Session . . . . .	58
Manipulating the Placement of the Application Window . . . . .	59
Managing Documents . . . . .	60
Working with the Current Drawing . . . . .	61
Creating and Opening Drawings . . . . .	61
Saving and Closing Drawings . . . . .	63
Accessing Information about a Drawing . . . . .	66
Manipulating a Drawing Window . . . . .	67
Working with System Variables . . . . .	68
Querying and Setting Application and Document Preferences . . . . .	70
Executing Commands . . . . .	71
Exercise: Setting Up a Project . . . . .	72
Creating the <i>DrawingSetup</i> Project . . . . .	73
Creating and Saving a New Drawing from Scratch . . . . .	74
Inserting a Title Block with the <i>insert</i> Command . . . . .	76
Adding Drawing Properties . . . . .	78
Setting the Values of Drafting-Related System Variables and Preferences . . . . .	80
<b>Chapter 4 • Creating and Modifying Drawing Objects . . . . .</b>	<b>83</b>
Understanding the Basics of a Drawing-Based Object . . . . .	83
Accessing Objects in a Drawing . . . . .	88
Working with Model or Paper Space . . . . .	89
Creating Graphical Objects . . . . .	91
Adding Straight Line Segments . . . . .	91
Working with Curved Objects . . . . .	92
Working with Polylines . . . . .	96
Getting an Object in the Drawing . . . . .	99
Modifying Objects . . . . .	101
Deleting Objects . . . . .	102
Copying and Moving Objects . . . . .	102
Rotating Objects . . . . .	103

Changing Object Properties . . . . .	104
Exercise: Creating, Querying, and Modifying Objects . . . . .	105
Creating the <i>DrawPlate</i> Project . . . . .	105
Creating the Utilities Class . . . . .	106
Defining the <i>CLI_DrawPlate</i> Function . . . . .	108
Running the <i>CLI_DrawPlate</i> Function . . . . .	110
Exporting the Utilities Class . . . . .	111

## **Chapter 5 • Interacting with the User and Controlling**

<b>the Current View . . . . .</b>	<b>113</b>
Interacting with the User . . . . .	113
Requesting Input at the Command Prompt . . . . .	114
Providing Feedback to the User . . . . .	125
Selecting Objects . . . . .	127
Selecting an Individual Object . . . . .	127
Working with Selection Sets . . . . .	129
Filtering Objects . . . . .	132
Performing Geometric Calculations . . . . .	134
Calculating a Coordinate Value . . . . .	134
Measuring the Distance Between Two Points . . . . .	135
Calculating an Angle . . . . .	136
Changing the Current View . . . . .	137
Zooming and Panning the Current View . . . . .	137
Working with Model Space Viewports . . . . .	139
Creating and Managing Named Views . . . . .	142
Applying Visual Styles . . . . .	143
Exercise: Getting Input from the User to	
Draw the Plate . . . . .	143
Revising the <i>CLI_DrawPlate</i> Function . . . . .	144
Revising the <i>Utilities</i> Class . . . . .	147
Using the Revised <i>drawplate</i> Function . . . . .	149

## **Chapter 6 • Annotating Objects . . . . . 151**

Working with Text . . . . .	151
Creating and Modifying Text . . . . .	151
Formatting a Text String . . . . .	153
Controlling Text with Text Styles . . . . .	156
Dimensioning Objects . . . . .	158
Creating Dimensions . . . . .	158
Formatting Dimensions with Styles . . . . .	160
Assigning a Dimension Style . . . . .	162
Creating and Modifying Geometric Tolerances . . . . .	163
Adding Leaders . . . . .	164
Working with Multileaders . . . . .	164
Creating and Modifying Legacy Leaders . . . . .	167
Organizing Data with Tables . . . . .	168

Inserting and Modifying a Table . . . . .	168
Formatting Tables . . . . .	169
Assigning a Table Style . . . . .	170
Creating Fields . . . . .	170
Exercise: Adding a Label to the Plate . . . . .	171
Revising the <i>CLI_DrawPlate</i> Function . . . . .	171
Revising the <i>Utilities</i> Class . . . . .	173
Using the Revised <i>drawplate</i> Function . . . . .	173
<b>Chapter 7 • Working with Blocks and External References . . . . .</b>	<b>175</b>
Managing Block Definitions. . . . .	175
Creating a Block Definition . . . . .	176
Adding Attribute Definitions . . . . .	178
Modifying and Redefining a Block Definition . . . . .	181
Determining the Type of Block Definition . . . . .	182
Inserting and Working with Block References . . . . .	183
Inserting a Block Reference . . . . .	183
Modifying a Block Reference . . . . .	184
Accessing the Attributes of a Block . . . . .	187
Working with Dynamic Properties . . . . .	189
Managing External References . . . . .	192
Working with Xrefs . . . . .	192
Attaching and Modifying Raster Images . . . . .	197
Working with Underlays . . . . .	199
Listing File Dependencies. . . . .	201
Exercise: Creating and Querying Blocks . . . . .	202
Creating the <i>RoomLabel</i> Project . . . . .	203
Creating the <i>RoomLabel</i> Block Definition . . . . .	203
Inserting a Block Reference Based on the <i>RoomLabel</i> Block Definition . . . . .	205
Prompting the User for an Insertion Point and a Room Number . . . . .	206
Adding Room Labels to a Drawing . . . . .	208
Creating the <i>FurnTools</i> Project . . . . .	209
Moving Objects to Correct Layers . . . . .	210
Creating a Basic Block Attribute Extraction Program . . . . .	212
Using the Procedures of the <i>FurnTools</i> Project . . . . .	219
<b>Chapter 8 • Outputting Drawings . . . . .</b>	<b>221</b>
Creating and Managing Layouts . . . . .	221
Creating a Layout . . . . .	222
Working with a Layout . . . . .	222
Controlling the Display of Layout Tabs . . . . .	223
Displaying Model Space Objects with Viewports . . . . .	223
Adding a Floating Viewport . . . . .	224
Setting a Viewport as Current . . . . .	225
Modifying a Floating Viewport . . . . .	225
Controlling the Output of a Layout . . . . .	228

Creating and Managing Named Page Setups . . . . .	229
Specifying an Output Device and a Paper Size . . . . .	229
Setting a Plot Style as Current . . . . .	232
Defining the Area to Output . . . . .	234
Changing Other Related Output Settings . . . . .	235
Plotting and Previewing a Layout . . . . .	235
Exporting and Importing File Formats . . . . .	237
Exercise: Adding a Layout to Create a Check Plot . . . . .	238
Creating the Layout . . . . .	239
Adding and Modifying a Plot Configuration . . . . .	240
Inserting a Title Block . . . . .	241
Displaying Model Space Objects with a Viewport . . . . .	242
Putting It All Together . . . . .	242
Testing the CheckPlot Procedure . . . . .	246

## **Chapter 9 • Storing and Retrieving**

<b>Custom Data . . . . .</b>	<b>247</b>
Extending Object Information . . . . .	247
Working with Xdata . . . . .	248
Defining and Registering an Application Name . . . . .	249
Attaching Xdata to an Object . . . . .	249
Querying and Modifying the Xdata Attached to an Object . . . . .	252
Removing Xdata from an Object . . . . .	258
Selecting Objects Based on Xdata . . . . .	258
Creating and Modifying a Custom Dictionary . . . . .	259
Accessing and Stepping through Dictionaries . . . . .	260
Creating a Custom Dictionary . . . . .	262
Storing Information in a Custom Dictionary . . . . .	263
Managing Custom Dictionaries and Entries . . . . .	264
Storing Information in the Windows Registry . . . . .	265
Creating and Querying Keys and Values . . . . .	265
Editing and Removing Keys and Values . . . . .	267
Exercise: Storing Custom Values for the Room Labels Program . . . . .	268
Attaching Xdata to the Room Label Block after Insertion . . . . .	269
Revising the Main <i>RoomLabel</i> Procedure to Use the Windows Registry . . . . .	269
Testing the Changes to the <i>RoomLabel</i> Procedure . . . . .	272
Persisting Values for the Room Label Procedure with a Custom Dictionary . . . . .	273
Retesting the <i>RoomLabel</i> Procedure . . . . .	275
Selecting Room Label Blocks . . . . .	276

## **Chapter 10 • Modifying the Application and Working with Events . . . . 279**

Manipulating the AutoCAD User Interface . . . . .	279
Managing Menu Groups and Loading Customization Files . . . . .	280
Working with the Pull-Down Menus and Toolbars . . . . .	281
Controlling the Display of Other User-Interface Elements . . . . .	293

Using External Custom Programs . . . . .	294
Working with Events . . . . .	295
Exercise: Extending the User Interface and Using Events . . . . .	300
Loading the <i>acp.cuix</i> File . . . . .	301
Specifying the Location of DVB Files . . . . .	302
Adding the Document Events . . . . .	303
Implementing an Application Event . . . . .	304
Defining the <i>AcadStartup</i> Procedure . . . . .	305
Testing the <i>AcadStartup</i> Procedure . . . . .	306
Testing the Application and Document Events . . . . .	307
<b>Chapter 11 • Creating and Displaying User Forms . . . . .</b>	<b>309</b>
Adding and Designing a User Form . . . . .	309
Adding a User Form to a VBA Project . . . . .	309
Considering the Design of a User Form . . . . .	310
Placing and Arranging Controls on a User Form . . . . .	312
Placing a Control on a User Form . . . . .	312
Deciding Which Control to Use . . . . .	313
Grouping Related Controls . . . . .	316
Managing Controls on a User Form . . . . .	317
Changing the Appearance of a User Form or Control . . . . .	319
Defining the Behavior of a User Form or Control . . . . .	321
Displaying and Loading a User Form . . . . .	324
Showing and Hiding a User Form . . . . .	324
Loading and Unloading a User Form . . . . .	325
Exercise: Implementing a User Form for the <i>DrawPlate</i> Project . . . . .	326
Adding the User Form . . . . .	326
Adding Controls to the User Form . . . . .	327
Displaying a User Form . . . . .	330
Implementing Events for a User Form and Controls . . . . .	331
Testing the User Form and Controls . . . . .	336
<b>Chapter 12 • Communicating with Other Applications . . . . .</b>	<b>339</b>
Referencing a Programming Library . . . . .	339
Creating and Getting an Instance of an Object . . . . .	340
Creating a New Instance of an Object . . . . .	341
Getting an In-Memory Instance of an Object . . . . .	344
Accessing a Drawing File from outside of AutoCAD . . . . .	346
Working with Microsoft Windows . . . . .	347
Accessing the Filesystem . . . . .	348
Manipulating the Windows Shell . . . . .	353
Using the Win32 API . . . . .	355
Reading and Writing Text Files . . . . .	356
Opening and Creating a File . . . . .	356
Reading Content from a File . . . . .	358

Writing Content to a File . . . . .	359
Closing a File . . . . .	360
Parsing Content in an XML File . . . . .	360
Working with Microsoft Office Applications . . . . .	363
Exercise: Reading and Writing Data . . . . .	365
Creating Layers Based on Data Stored in a Text File . . . . .	366
Searching for a File in the AutoCAD Support Paths . . . . .	369
Adding Layers to a Drawing with the <i>LoadLayers</i> Procedure . . . . .	370
Writing Bill of Materials to an External File . . . . .	371
Using the <i>FurnBOMExport</i> Procedure . . . . .	374
<b>Chapter 13 • Handling Errors and Deploying VBA Projects . . . . .</b>	<b>375</b>
Catching and Identifying Errors . . . . .	375
Recovering and Altering Execution after an Error . . . . .	375
Getting Information About the Recent Error . . . . .	378
Debugging a VBA Project . . . . .	381
Debugging Through Messages . . . . .	381
Using the VBA Editor Debug Tools . . . . .	383
Deploying a VBA Project . . . . .	388
Loading a VBA Project . . . . .	388
Specifying the Location of and Trusting a Project . . . . .	392
Starting a Macro with AutoLISP or a Command Macro . . . . .	394
Grouping Actions into a Single Undo . . . . .	395
Protecting a Project . . . . .	396
Exercise: Deploying the DrawPlate VBA Project . . . . .	396
Stepping Through the BadCode VBA Project . . . . .	397
Implementing Error Handling for the Utility Procedures . . . . .	399
Implementing Error Handling and Undo Grouping for the Main Procedures . . . . .	401
Configuring the AutoCAD Support and Trusted Paths . . . . .	405
Creating <i>DrawPlate_VBA.bundle</i> . . . . .	405
Deploying and Testing <i>DrawPlate_VBA.bundle</i> . . . . .	406
<i>Index</i> . . . . .	409



# 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](http://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](http://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](mailto: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](http://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](http://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 .dwb 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
```