

Bonus Chapter 2: The AutoCAD Programming Interfaces

In This Chapter

- ✓ Finding out what programming AutoCAD can do for you
- ✓ Covering the available programming interfaces
- ✓ Mastering the ins and outs of the programming interfaces
- ✓ Figuring out which programming interface is best for you

Have you ever wanted to create a command alias that represented a specific option of a command? Or maybe you have established CAD standards, but some of the settings cannot be set up through a drawing template alone, so you want to make sure that the settings are set before any work on a design begins? If so, you have come to the right place — or section of the book. AutoCAD is more than just a drafting tool. Although customizing AutoCAD can help increase productivity by itself, the programming interfaces allow you to tap into much more powerful resources that are contained in the depths of AutoCAD. (Okay, saying the resources are in the depths of AutoCAD might be a bit of an exaggeration because they are used every time you run a command from the AutoCAD interface.)



This chapter is aimed at AutoCAD users only. Sorry AutoCAD LT users — the programming interfaces are limited to AutoCAD.

The supported application programming interfaces (APIs) for AutoCAD are available after you install AutoCAD or after you've downloaded them from the Autodesk Web site. APIs are used to communicate with AutoCAD, any open drawing files, and the objects in a drawing. With some of the APIs, you can create your own custom commands that users can execute from a command prompt or by using an element in the user interface. The power behind some of these programming interfaces is that you don't need to be a programmer to take advantage of them.

Discovering What You Can Do by Programming AutoCAD

You might be thinking to yourself, “I’m not a programmer, so why do I want to know about programming interfaces?” The best reason why you should understand the programming interfaces is to simplify repetitive tasks in your drafting workflow. If you can create basic custom programs that save 15 minutes a day, the effort is worth your time — especially if you can share these programs with your coworkers.

Over time, those 15 minutes can grow into much more as you become more efficient with the programming interfaces. Custom programs don’t need to be complex to increase efficiency. They can be simple, such as creating new commands that perform a Zoom Previous from the command prompt. A complicated custom program might take on issues such as CAD standards.

Managing CAD standards can be a nightmare, but the process can be improved by using the available programming interfaces. Tasks such as making sure dimensions are placed on the correct layer can be accomplished if you understand the programming interfaces and how AutoCAD works. You don’t need to understand how the information is actually written to the file, but you do need to know how it is logically organized. (By *logically organized*, I mean you should understand that objects such as layers are not just floating around inside a drawing; instead, they are stored in a table that contains all the layers in a drawing.)

The advantages of using APIs

The advantages of APIs differ based on whether you are already using third-party applications or add-ons for AutoCAD. Even if you are, you may still discover advantages to in-house programming. Here are some of the benefits for creating custom programs for AutoCAD:

- ◆ **Accuracy:** If a process has a large number of steps, some might be overlooked, causing errors to creep into a design. By creating a custom program that runs consistently every time, you can increase the accuracy of your drawings.
- ◆ **Appearance:** Programming can aid in making drawings look uniform by allowing you to set up drawing options that cannot be defined in a drawing template or by allowing you to automate the updating of objects in a drawing.
- ◆ **Efficiency:** Repetitive tasks can be sped up, enabling drafters to spend more time on the design process instead of performing the same set of steps over and over again to finish a design.

- ◆ **Training efficiency:** Training new employees is always a challenge. If a complex or large set of processes must be followed, it can take new employees longer to be productive. Wrapping custom programs around processes can help new employees become productive more quickly.
- ◆ **The downstream effect:** Being able to get things done faster and more efficiently during the drafting process is great, but do not forget about those people who use the data from a design after it has been completed. Custom programming can be used to extract information out of a drawing or set of drawings that can be useful downstream in manufacturing, sales and marketing, and many other areas of a company.

The other side of the story

There are always two sides to every story. I have covered some of the benefits of using the programming interfaces in AutoCAD, but there have to be downsides, right? There are, and here are some of the disadvantages to programming in AutoCAD:

- ◆ **Cost:** Programming in AutoCAD costs money — how much depends on which programming interface you go with. These costs might be in the form of software, time, or consulting fees. In the case of a few of the programming interfaces, no additional software packages need to be purchased but, regardless, time is still a cost in these types of projects.
- ◆ **Maintenance:** AutoCAD changes from release to release. When you upgrade to a new version, you may need to spend time updating your custom programs. Some feature that your custom program uses may change, or you may want to take advantage of a new feature that improves a process even further.
- ◆ **Learning curve:** Even though a number of the programming interfaces are easy for nonprogrammers to understand, they still require you to put in some time if you want to become proficient. So before you promise your boss that you can deliver improved productivity, make sure you are comfortable with the programming language.

Getting to Know the Available Programming Interfaces

As AutoCAD has evolved over the years, so have the different programming interfaces that are available. When AutoCAD was introduced to the world, no programming interfaces were built into the application. After about three years, the first programming interface, AutoLISP, was added. AutoLISP, which was based on the LISP programming language, allowed AutoCAD users to tailor the program to the way they wanted to work.

After AutoLISP came the introduction of ADS (AutoCAD Development System), which introduced C-style coding as a programming option. ADS had a short life span because the C language was already being overshadowed by the next generation of the C programming language, C++. ADS, which was only around for about three years, evolved into ObjectARX, which is still the premier programming option in AutoCAD 2009. If you look at the install directory of AutoCAD, the number of files with the ARX and DBX file extensions indicate that ObjectARX is the tool of choice by Autodesk itself to extend the core functionality of AutoCAD.

In 1997, Autodesk added the ActiveX programming interface, which allowed Visual Basic (VB) programmers to extend the core functionality of AutoCAD. ActiveX is not just for VB programmers; many other mainstream languages such as Java and C++ support ActiveX so they can interface with AutoCAD. Autodesk didn't stop with ActiveX support. In recent years, Autodesk has started to focus more on implementing programming libraries for use with the .NET programming languages developed by Microsoft and other vendors.

AutoLISP

As mentioned, AutoLISP is a programming language based on the LISP language. LISP stands for list processing. It was introduced back in 1958 and was popular during the '70s and '80s. Programmers often joke that the LISP acronym really stands for Lost In Stupid Parentheses due to the use of parentheses to delimit the start and end of an expression. LISP uses open and closed parentheses for an expression, as the following example shows:

```
(command "line" "0,0" "5,5" "")
```

The example uses the LINE command to draw a line starting at the coordinate 0,0 and ending at 5,5. As you can see, it's not much different from what you type at a command prompt.

AutoLISP can be used to organize multiple expressions into a custom command that a user can enter at a command prompt or use in a command macro for a toolbar button. To create a new command or function that can be used to extend the built-in AutoLISP functions, you use the function DEFUN — DEFINE FUNCTION.

Although Autodesk no longer uses AutoLISP as its primary programming language, many users (after they get past all the parentheses) still use it to automate repetitive tasks. As a whole, the programming interface is by far one of the most cost-effective and forgiving of the four programming interfaces I discuss in this book.

The last major update to the AutoLISP programming language was back in 1998 when Autodesk purchased a package called Vital LISP. Vital LISP was then renamed Visual LISP and sold as an add-on for AutoCAD 14. Visual LISP was then included as part of AutoCAD 2000, which was released in 1999. The Visual LISP environment extended the functionality of the AutoLISP language by adding hundreds of new functions and allowing AutoLISP to access the AutoCAD Object Library similar to VBA and ActiveX.



The AutoCAD Object Library allows you to create and modify objects contained in an AutoCAD drawing file. The AutoCAD Object Model is a visual representation of the AutoCAD Object Library and can be used to locate and access the various objects in the ActiveX programming interface for AutoCAD.

ActiveX automation

ActiveX automation is also known as Component Object Model (COM) automation. COM automation is a form of component-based software architecture that allows an application to expose its internal functionality in the form of objects. COM allows applications to cross-communicate to exchange information or interact. Many modern programming languages, such as VB/VBA, C++, and Java, can use COM automation.

So why is ActiveX automation important in AutoCAD? ActiveX automation allows you to use a rich and modern programming language such as VB or VBA to communicate with AutoCAD, which means you can exchange information with data sources such as a Microsoft Access database or even a Microsoft Excel spreadsheet. Exchanging information with data sources can allow you to improve downstream processes by providing CAD information to non-CAD users who might use the information for billing or manufacturing.

One of the advantages to using ActiveX automation is that you are not limited to just building applications in AutoCAD. If you want to develop a standalone application with VB and communicate with AutoCAD, you can. If you create a VBA project in Microsoft Word and want to communicate with AutoCAD, you can do that too. In Bonus Chapters 3 through 5, I discuss ActiveX and VBA as one and the same because VBA uses ActiveX automation.

VBA

VBA — Visual Basic for Applications — has been around for some time and is an extension of the popular programming language Visual Basic (VB). VBA is defined as an object-oriented programming (OOP) language. The concept behind OOP is that a computer program is developed by using a collection of individual units, or *objects*, as opposed to a listing of instructions. Each object has the ability to receive messages, process data, and transmit messages to other objects.

VBA is part of the programming and development tools developed by Microsoft. It dates back to MS-DOS and programming languages such as QBasic and MS-Basic. VB has been around longer than VBA and, unlike VBA, VB can be used to build standalone applications that are not dependent on a host application.



A *host application* is a program that allows the VBA environment to run inside it; an example of this is AutoCAD. Many popular Windows-based programs have VBA technology built into them. Some of the applications with VBA built inside are Autodesk Inventor, AutoCAD-based vertical products such as AutoCAD Architecture (formally known as Autodesk Architectural Desktop), and Microsoft Word and Excel.

VBA in AutoCAD has been a welcomed feature from both the development and nondevelopment communities. This programming option allows for the integration of business applications directly into the AutoCAD environment and allows companies to tap into an existing developer community that already knows VB/VBA.

ObjectARX and ObjectDBX

ObjectARX and ObjectDBX are not programming languages like VBA, but instead are programming interfaces that allow developers to create and extend AutoCAD by using the object-oriented C++ programming language. ObjectARX is a collection of library and include files that provide you with versatile tools that can extend the core functionality of AutoCAD and other AutoCAD-based products such as AutoCAD Architecture (formally known as Autodesk Architectural Desktop) and AutoCAD Mechanical.

The AutoCAD-based vertical products are a collection of ObjectARX programs that are focused on a specific target audience and the type of work they perform. ObjectDBX files are used to define custom drawing objects like the ones found in AutoCAD Architecture rather than user interface elements and commands like those in ObjectARX files. If your programs are calculation intensive, you might want to look at using ObjectARX for these types of tasks instead of using AutoLISP or ActiveX.

ObjectARX allows for smaller, compact files, faster execution, and tighter integration with AutoCAD and Windows than the other programming options allow. Unlike AutoLISP and VBA, ObjectARX requires you to purchase additional development tools and download the software development kit from the Autodesk Web site. When purchasing development tools, you are limited to the version that was used to build the AutoCAD release for which you are creating ObjectARX and ObjectDBX programs. To know which version of C++ you need to work with ObjectARX, you should consult the online Help that comes with the ObjectARX software development kit (SDK). The ObjectARX SDK can be downloaded from <http://www.objectarx.com>.

.NET

.NET is Microsoft's alternative to portable programming languages such as Java and J2EE and represents Microsoft's latest generation of programming languages that offer flexibility for both application development and usability. To make .NET appealing to the development community, the Visual Studio .NET development environment incorporates more than twenty languages, such as RPG, COBOL, and C#.

The .NET programming interface is similar to the basic concept of ActiveX automation but with much greater flexibility through the use of newer programming languages and technologies. Like ObjectARX, a separate development environment must be obtained; it isn't supplied with AutoCAD. However, like ActiveX automation, the .NET API is available upon installing AutoCAD. .NET help and samples are available as part of the ObjectARX software development kit and must be downloaded from the Autodesk Web site. The ObjectARX SDK can be downloaded from <http://www.objectarx.com>.

Comparing Strengths and Weaknesses of the Programming Interfaces

Now that you have an idea of the programming interfaces available to use with AutoCAD, you will want to see how they stack up. Here, I break down some key areas to help you understand how the different programming interfaces measure up against each other:

- ◆ **Learning curve:** If you are not experienced with programming and do not have much time to dedicate to mastering a programming language, AutoLISP is the best option for you. It is powerful but does not require you to master a lot of different concepts to get a basic program together. If you do have a background in programming, you may want to look at using VBA in AutoCAD because it is based on VB, which is a programming language that is not specific to AutoCAD.

If you have a considerable amount of programming experience, you might consider using ObjectARX or .NET, which take longer to master but are powerful. (.NET has a smaller learning curve compared to ObjectARX.)

- ◆ **Execution speed:** Both AutoLISP and VBA use interpreters to talk to AutoCAD, which can cause lag in execution and overall execution speed. ObjectARX is by far the fastest and most efficient of all the programming interfaces. .NET, although flexible, uses Just In Time (JIT) compiling, which can cause an initial performance lag the first time the program is run.

BC20 Comparing Strengths and Weaknesses of the Programming Interfaces

- ◆ **Cost:** AutoLISP and VBA are both built into AutoCAD, so no additional development tools must be purchased. ObjectARX and .NET require the purchase of additional tools that can range in price from hundreds to thousands of dollars. This might affect which programming interface you use first.
- ◆ **User input:** All the programming interfaces allow you to get information from a user, but how a user's input is collected varies slightly. AutoLISP has better command prompt support than VBA because AutoLISP allows you to create custom commands and VBA doesn't. If you are looking to offer a graphical interface for your programs, you might want to look at VBA instead of AutoLISP.

AutoLISP uses DCL (Dialog Control Language) to define how a dialog box should look. This is different from the other programming languages, which use a graphical editor to design the look of dialog boxes that are used with a custom program.

ObjectARX has the best support in this area; but it also has a number of additional things you have to contend with, such as Microsoft Foundation Classes (MFC), which are tools that help you create robust dialog boxes and user interfaces. .NET is flexible in gathering user input, but unlike ObjectARX, it lacks some control over UI elements specific to AutoCAD.

- ◆ **Maintenance:** AutoLISP and VBA are the easiest to maintain because they typically run on future releases with few changes required. ObjectARX programs typically need to be recompiled about every third release of AutoCAD due to binary compatibility issues with the library files. .NET is the newest programming interface that Autodesk offers for use with AutoCAD, and although it is starting to become a mature programming option, a future release of AutoCAD might force you to rebuild your custom programs. In the long term, I hope the .NET API follows suit with AutoLISP and VBA to reduce the amount of maintenance between releases.
- ◆ **Longevity:** AutoLISP has been around the longest and is the easiest of the four programming interfaces to master. VBA is newer to AutoCAD than AutoLISP, but it is nearing the end of its life cycle according to Microsoft. (This is due to the fact that ActiveX is being replaced by .NET, which is much newer and more flexible.) Keep in mind that just because VBA is getting close to the end of its life cycle does not mean that support will suddenly stop; it will just be slowed down.

ObjectARX and .NET are probably the APIs most likely to be around for a while because Autodesk uses them to introduce new features. Although both ObjectARX and .NET seem to have bright futures, no one ever knows for sure where technology will go.

Deciding Which Programming Interface Is Best for You

So which of the four programming interfaces might be best for you? The two that most users find easiest to use and master are AutoLISP and VBA, which is why I cover them in this book. But just because AutoLISP and VBA are covered in this book does not necessarily make them the right choices for you. To determine which option is best for you, read the following descriptions of each programming interface and make a decision based on which description best matches your scenario.

AutoLISP is a good fit if you know AutoCAD and the commands that are available to you pretty well or if you do not have much or any experience in programming. Most first-time AutoLISP programmers feel that it is a much more natural transition into programming because they can use the commands they are already familiar with to automate tasks. It also allows for plenty of growth because it can communicate with other applications by using the Visual LISP functions via ActiveX automation. AutoLISP is the easiest for beginners to master and work with.

Typically, VBA is a good fit if you do not know much about AutoCAD but have a background in VBA or VB programming. VBA enables companies with IT departments to get involved with extending AutoCAD because a large number of programmers know VB. VBA is a good starting point for beginners, but because you are not using AutoCAD commands, the learning curve is higher. VBA is a programming interface that both beginners and intermediate users can work with.

.NET is the middle-of-the-road option between VBA and ObjectARX. Although it requires you to understand one of the newest programming languages, such as VB.NET or C#, it is not as complex to master as ObjectARX and C++. Although the .NET programming interface does have some limitations in its current implementation, it is improving with each new release. .NET provides some of the simplicity of application development that comes with VBA but has most of the power of C++ with a smaller learning curve. Although VB.NET is different from VBA, the syntaxes look and feel similar. One of the important things that the two share is the capability to use ActiveX automation. You can start with VBA and, over time, transition to VB.NET.

ObjectARX should typically be left to those who have experience with C++ because a lot of things can go wrong in a hurry if you don't understand the C++ programming language. In C++, you are responsible for managing memory, so if you make a mistake, you can develop programs that bring AutoCAD to its knees quickly. If you can tame the C++ programming language beast, you will have just about everything you can imagine at your fingertips for creating custom programs in AutoCAD.

