Chapter 1: 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 can't be set up through a drawing template alone. So you want to make sure the settings are finished before any work on a design begins? If so, you have come to the right place. 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 since they are used every time you run a command in the AutoCAD interface.)

This minibook 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 the AutoCAD application, any open drawing files, and objects in a drawing. With some of the APIs, you can create your own custom commands that users can execute from the command line. 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 to discover the programming interfaces is to simplify repetitive tasks in your workflow. If you can create very 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, your 15-minute savings can grow into much more as you become more efficient with the programming interface. Custom programs don't need to be complex to increase efficiency. They can be simple, like creating new commands that can be used to perform a Zoom Previous from the command line, or complicated, taking on issues such as CAD standards.

Managing CAD standards can be a nightmare, but the process can be improved by using 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, we mean you should understand that objects such as layers aren't 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:** By creating a custom program that runs consistently every time, you can increase the accuracy of your drawings. If a process has a large number of steps in it, some steps might be overlooked. This can cause errors to creep into your design.
- **Appearance:** Programming can aid in making drawings look uniform by allowing you to set up drawing options that can't be defined in a drawing template or by allowing you to automate the updating of objects in a drawing.
- **Efficiency:** Repetitive tasks can be speeded up, enabling drafters to spend more time on the design process.

- **Training efficiency:** Training a new employee is always a challenge. If a complex or large set of processes must be followed, new employees take even longer to be productive. Wrapping custom programs around processes can help get them up and productive in a shorter period of time.
- **The downstream effect:** Being able to get things done faster and more efficiently during the drafting process is great, but don't forget about those people who use the data after the design 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.

The other side of the story

There are always two sides to every story. We've covered some of the benefits of using the programming interfaces in AutoCAD, but there has to be a downside, right? There is, 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 couple of the programming interfaces, no additional software packages need to be purchased, but, regardless, time is always a cost in these types of projects.
- **Maintenance:** AutoCAD changes from release to release. When you upgrade to a new version, you often have to spend time updating your custom programs. Some feature your custom program was using may get broken, or you may want to take advantage of a new feature that is introduced to improve a process even further.
- **Learning curve:** Even though a number of the programming interfaces are easy for non-programmers to pick up and 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 you chose first.

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 first introduced to the world, no programming interfaces were built into the application. Only about three years after the product was released was the first programming interface added to the software. This programming interface was AutoLISP, which

was based on the LISP programming language. The introduction of AutoLISP allowed users of AutoCAD to tailor the program for how they wanted to work to improve productivity.

After AutoLISP came the introduction of ADS (AutoCAD Development System), which introduced C-style coding as a programming option. ADS had a rather 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 2007. If you look at the install directory of AutoCAD, you can see that this is the tool of choice by Autodesk itself to extend the core functionality of AutoCAD.

In 1997, Autodesk extended the programming interfaces it was offering to include ActiveX, which allowed 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. It kept up with the ever-changing landscape of technology and introduced a programming interface for .NET that could be used with the new .NET languages by Microsoft and other development language vendors.

AutoLISP

AutoLISP is a programming language based on the LISP language. LISP stands for *lis*t *p*rocessing. It was first introduced back in 1958 and was popular during the '70s and '80s. Programmers often joke that the LISP acronym really stands for *L*ost *I*n *S*tupid *P*arentheses. Take one look at the code, and you might agree. LISP uses start and end parentheses for a statement, as the following example shows:

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

In the example, you can see the use of the parentheses to start and end the expression. 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 are use to typing in at the command line. Although not a very powerful statement, it is rather easy to understand.

AutoLISP can be used to organize multiple statements into a custom command that a user can enter at the command line or use in a command macro for a toolbar button. To create a new command or a function that can be used to extend the built-in AutoLISP functions, use the function DEFUN — DEfine FUNction.

Although AutoLISP is no longer used as the primary development tool by Autodesk as it once was, it is still the primary tool that many users work with because it is easy to learn after you get past all the parentheses. As a whole, the programming interface is by far one of the most cost-effective and forgiving of the four different programming interfaces we 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 to Visual LISP and sold as an add-on originally for AutoCAD 14. Visual LISP was then included as part of the AutoCAD 2000 release in 1999. The Visual LISP environment extended the functionality of the AutoLISP language by adding hundreds of functions and allowing AutoLISP to access the AutoCAD Object Model similar to VBA and ActiveX.

The AutoCAD Object Model is the documented structure of the relationships between AutoCAD and other objects like lines and blocks that are contained in an AutoCAD Drawing file. The object model is used as a road map to locate and access the various different 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 with each other to exchange information or interact. Many of the modern programming languages like VB/VBA, C++, and Java are capable of using COM automation.

So why is ActiveX automation important in AutoCAD? ActiveX automation allows you to use a rich and modern programming language like VB or VBA. When you can use a programming language like VB or VBA when communicating with AutoCAD, you can exchange information with data sources like an MS Access database or even an MS Excel spreadsheet. This can allow you to improve downstream processes by providing CAD information in different systems 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 MS Word and want to communicate with AutoCAD, you can do that too. In the remainder of this minibook, we 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 very 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 using a collection of individual units, or *objects,* as opposed to a listing of instructions. Each one of the objects has the ability to receive messages, process data, and transmit messages to other objects.

VBA is part of the programming and development tools that are developed by Microsoft. Its roots date back to MS-DOS and programming languages like 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 other applications are Autodesk Inventor, AutoCAD-based vertical products such as Architectural Desktop (ADT), and Microsoft Word and Excel.

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

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 using the object-oriented C++ programming language. ObjectARX is a collection of library and include files that provides you with the versatility of tools that Autodesk uses to extend the core functionality of AutoCAD and other AutoCAD-based products such as Autodesk Architectural Desktop (ADT) and AutoCAD Mechanical Desktop (MDT).

The 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 ADT, instead of user interface elements and commands like in ObjectARX files. If your programs are calculation-intensive, you might want to take a look at using ObjectARX for these types of tasks.

ObjectARX allows for smaller or more compact files, faster execution, and tighter integration into AutoCAD and Windows. Unlike the other programming options, with ObjectARX, you need to purchase additional development tools and download the software development kit from the Autodesk Web site. When purchasing the 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.

.NET

.NET is Microsoft's alternative to portable programming languages like Java and J2EE. .NET represents Microsoft's latest generation of the programming languages that offer flexibility for both application development and usability. To make .NET appealing to the development community, the development environment Visual Studio .NET incorporates more than 20 different languages, such as RPG, COBOL, and C#.

The .NET programming interface is very 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 development kit and must be downloaded from the Autodesk Web site.

Comparing Strengths and Weaknesses of the Programming Interfaces

Now that you have an idea of the programming interfaces that are available to use with AutoCAD, you'll want to see how they stack up. Here, we 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 don't have much time to dedicate to mastering a programming language, AutoLISP is the best option for you. It is very powerful but doesn't require you to learn 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 a universal programming language.

  If you have a considerable amount of programming experience, you might consider using ObjectARX or .NET, which take longer to learn but are very powerful. (.NET has far fewer pain points compared to ObjectARX.)

- **Execution speed:** Both AutoLISP and VBA use interpreters to talk to AutoCAD, which can cause a lag in execution time and overall execution speed. ObjectARX is by far the fastest and most efficient of all the different 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.
- **Cost:** AutoLISP and VBA are both built in to AutoCAD, so no additional development tools must be purchased. ObjectARX and .NET require the purchasing 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 of the different programming interfaces allow you to get information from a user, but how a user's input is collected varies slightly. AutoLISP has better command line support over VBA because AutoLISP allows you to create custom commands and VBA doesn't. If you are looking to offer dialog boxes in your programs, you might want to look at VBA over AutoLISP.

  AutoLISP uses a language called DCL (Dialog Control Language) that defines 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.

  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 very flexible in gathering user input, but, unlike ObjectARX, it lacks some control over UI elements that are specific to AutoCAD.
- **Maintenance:** AutoLISP and VBA are the easiest to maintain as they typically work on future releases with few changes required. ObjectARX programs typically need to be rebuilt about every third release of AutoCAD due to binary compatibility issues with the library files. .NET is a young programming interface for AutoCAD. Until it matures, you can almost guarantee that any custom programs will need to be rebuilt to some degree between releases. In the long term, we hope that .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 learn. 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 doesn't 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 they are the ones used by Autodesk. ObjectARX seems to have a very bright future, but 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 is best for you? The two that most users find easiest to use and learn are AutoLISP and VBA, which is why we cover them in this book. But just because AutoLISP and VBA are covered in this book doesn't 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 most closely 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 don't 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 that they are already familiar with to automate tasks. It also allows for plenty of growth because it can communicate with other applications using the Visual LISP functions via ActiveX automation. AutoLISP is the easiest for beginners to learn and work with.

Typically, VBA is a good fit if you don't really know much about AutoCAD but have a background in VBA or VB programming. VBA enables companies with IS 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 newer programming languages, such as VB.NET or C#, it isn't as complex as mastering ObjectARX or 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 the power of C++ with a smaller learning curve. Although .NET is different from VBA, the syntaxes look and feel very similar to each other. One of the important things that the two share is the capability to use ActiveX automation. You could start with VBA and, over time, transition to .NET.

ObjectARX should typically be left up 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 your own 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.